Mining Intentional Process Models

Ghazaleh Khodabandelou

UNIVERSITY of Paris 1 PANTHEON-SORBONNE

DOCTORAL SCHOOL EMPS

Applied Mathematics and Informatics

Ph.D. T H E S I S

to obtain the title of

Ph.D. of Science

of the University of Paris 1 Panthéon-Sorbonne

Specialty: Computer Science

by

Ghazaleh KHODABANDELOU

Mining Intentional Process Models

Thesis Supervisor: Prof. Camille Salinesi

Thesis Co-supervisors: Dr. Charlotte Hug
and Dr. Rebecca Deneckère

prepared at Centre de Recherche en Informatique

defended on June 13, 2014

Jury:

Supervisor:	Prof. Camille Salinesi - University of Paris 1 Panthéon-Sorbonne
Co-supervisors:	Dr. Charlotte Hug - University of Paris 1 Panthéon-Sorbonne
	Dr. Rebecca Deneckère - University of Paris 1 Panthéon-Sorbonne
Reporters:	Prof. Judith Barrios - University of Los Andes - Venezuela
	Dr. HDR. Samira Si-said Cherfi - CNAM Paris
Examiner:	Prof. Régine Laleau - University of Paris-Est Créteil

Remerciement

Mes remerciements vont tout d'abord au professeur Camille Salinesi et aux docteurs Charlotte Hug et Rebecca Deneckère qui m'ont guidée et soutenue tout au long de cette thèse.

Je remercie le professeur Jane Cleland pour m'avoir accueillie dans une équipe formidable et pour ses conseils enrichissants durant mon séjour à Chicago. Je remercie également Mehdi Mirakhorli pour ses conseils très constructifs et sa présence.

Je remercie le professeur Judith Barrios et le docteur Samira Si-Saïd Cherfi pour leur conseils qui m'ont permis d'améliorer la qualité de cette thèse.

Je remercie tous les permanents du laboratoire de Centre de Recherches en Informatique, que j'ai eu le plaisir de côtoyer, notamment les professeurs Collette Rolland, Bénédicte Le Grand, Carine Souveyet. Je remercie Raul et Daniel pour leur soutien.

Je remercie le directeur du laboratoire des Signaux et Systèmes, Silviu Niculescu qui m'a toujours soutenue et m'a donné l'occasion de débuter cette thèse.

Je ne peux bien sûr pas oublier tous les doctorants et futurs docteurs du laboratoire : Sana, Cosmin, Amina, Salma, Dany, Ali.

Je remercie l'équipe de l'incubateur IncubAlliance, les sociétés Spraed, LiXoft, WiN MS, DeXXon et plus particulièrement François, Jakob, Eric, Laurent, Jean-François, Chatel et Armand pour leur collaboration.

Enfin, je remercie mes parents et ma sœur pour leur amour et leur présence indéfectible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Context

Process models play an important role in organizations. Process models are used for a variety of reasons when designing, redesigning, improving processes and introducing new information systems [Van der Aalst 2011c]. They are used to view the process from various angles, to structure stakeholders' discussions, to document, to find errors in systems or procedures, to understand the factors influencing response times, to configure a system, to play out different scenarios and thus provide feedback to the designer, etc [Van der Aalst 2011c].

Process modeling often follows a traditional top-down methodology, starting from the requirements, up to the configuration and redesigning [Van der Werf 2011]. The architecture of such systems also must evolve with the system. However, architecture and realization often do not concur. In other words, process models are not aligned with reality, *i.e.*, what really happens in processes; thus, they cannot provide valuable information for stakeholders. As a consequence, there is a gap between the implemented systems that support the processes and those that are enacted in the real world. Thus, the systems become difficult to maintain. Indeed, in most organizations the changes in the systems are often not documented in a systematic and continuous manner [Van der Aalst 2011c].

Nowadays, the impressive growth of event logs makes it possible to analyze them. These event logs recorded by information systems contain precious information about the actual enacted processes. They can be used to provide an insight into the actual processes; deviations can be analyzed, and the quality of models can be improved. However, most of the recorded event logs are not well-structured, which cause problems while exploiting such enormous quantities of data. One of

the main challenges in organizations is to extract information and value from event logs recorded in information systems [Van der Werf 2011, Van der Aalst 2011c].

Given the growth of event logs in organizations, the advantages of process modeling, and the restricted quality of hand-made process models, it seems advantageous to relate event logs produced while enacting the process to process models. Therefore, the actual processes can be discovered to understand what users really do and existing process models can be evaluated and enhanced. The **process mining** field has emerged to bridge the gap between event logs and process models [Van der Aalst 2011c]. Process mining idea initially appeared in the software engineering field with Cook and Wolf [Cook 1998a], and has been applied on workflow logs for the first time by Agrawal [Agrawal 1998]. Process mining establishes links between the actual processes and process models. This research discipline sits between machine learning and data mining on the one hand and process modeling and model analysis on the other hand [Van der Aalst 2011c]. The basic idea is to extract knowledge from event logs recorded by information systems that contain precious information about all the interactions of the users with information systems. Process mining approaches aim at modeling users' behaviors in terms of sequences of tasks and branching in an automatic way [Rozinat 2007, Van der Aalst 2004a, Van der Aalst 2011c]. Mining processes from logs can be useful for understanding how people really work (individually or collectively), analyzing how actual processes differ from the prescribed process models, and thereby improve the quality of the enacted processes and increase the productivity. However, process mining approaches are rigid because they only take into account the process from the activities and their relationships point of view. Indeed, due to the rigidities of process mining approaches, they encounter a number of issues such as noise, hidden tasks, duplicate tasks, loops, etc [Van der Aalst 2004c].

1.2 Problem Statement

So far, the mined process models are activity oriented models. According to Dowson [Dowson 1987], the process models can be classified into three groups of models: activity-oriented, product-oriented and decision-oriented models. Activity-oriented process models [Rolland 2005a] concentrate on the activities and tasks performed in producing artifacts and their ordering. Product-oriented process models [Rolland 2005a] are concerned about the successive product transformations. Decision-oriented process models [Jarke 1992] introduce the concept of reasoning, choice and decision-making, the processes are then seen as teleological [Veblen 1898, Ralph 2008]. A teleological process is a process that takes into account the teleological behaviors of process enactment (behaviors attached to the notion of goal). It describes the intentions (goals, objectives) associated with a result that an individual wants to obtain. Taylor proposes a classification of behaviors focused either on goal-oriented (teleological behaviors) or on the response to stimuli [Taylor 1964]. A teleological behavior is defined as follows: when a specific

behavior is needed to meet a goal, then this behavior occurs [Malcolm 1967]. Indeed, the behaviors can be changed during the process enactment to meet the goal, according to the situation changes. Teleological principles of Veblen [Veblen 1898] incorporate actors' intentions in the process, which defines a teleological process as *"a dynamic equilibrium in which the rules are determined endogenously, as a relationship between the behavior of the system and its intentions"*.

In the late 90s, Rolland introduced a new category for the process models, called **intentional process model** [Rolland 1999], which takes into account the notions of intention and strategy to model the process enactment. A strategy is an approach, a manner or a means to achieve an intention [Rolland 2007]. Specifying intentions and strategies has proved a powerful tool to better understand the deep nature of processes, to see how processes interweave and combine, to abstract processes and visualize them under man-manageable form, even when they are extremely complex [Rolland 2005b]. Intention-oriented process models have emerged to offer a flexible structure to process models. Many research works in intention-oriented process modeling demonstrate that the fundamental nature of processes is mostly intentional and the process should be modeled from an intentional point of view [Davis 1989, Plihon 1996, Rolland 1999]. According to these approaches, an enacted process is a reflection of humans' intention performed as a sequence of activities. Therefore, it is not possible to model humans' cognitive operators, *e.g.*, thinking, deciding, and acting process only in terms of a simple sequence of activities. Indeed, an intention is a goal that a user wants to achieve regarding the context in which he/she is working [Plihon 1996]. The notion of context plays a key role for the intention, since a given intention emerges in a given context, which not only promotes its appearance, but also influences the realization of this intention [Rolland 2005b]. In the method engineering context [Janković 2013], to understand methods used by stakeholders and their ways of working, it is essential to capture intentions that led to the implementation of activities.

Intentions are a first class concept of information systems engineering [Rolland 2005b]. In the early 80s, intention models have been proposed in information systems community [Swanson 1974, Christie 1981] as a *potential theoretical foundation* to determine user's behavior [Davis 1989]. Intention models take root in a former work Technology Acceptance Model (TAM) [Davis 1989] one of the extensions of Theory of Reasoned Action (TRA) [Ajzen 1975] designed to model humans' behavioral intention, specially for computer usage behavior. The TRA has proven effective in predicting and explaining humans' behavior through various domains. Since the early 90s, intention-oriented software process specification have been promoted as a driving paradigm to study strategic alignment [Thevenet 2007b, Etien 2006], to define actors and roles, to specify the outcome of business process models [Salinesi 2003] and name them, to analyze, to support guidance [Rolland 1993, Deneckère 2010], to describe intentional services [Rolland 2010], to handle traceability issues [Jarke 1993], to express pervasive information systems [Najar 2011], to define systems requirements [Ralyté 1999], to study users' behavior to identify and name use cases, to tailor methods [Ralyté 2003]

or to make more flexible methods [Mirbel 2006], etc. Further, research on guidance in method engineering shows that many method engineering issues, such as rigidity or lack of adaptation, are solved more effectively when intentions and strategies are explicitly specified [Rolland 2005b].

Many works on intention-oriented modeling indicate how to express them, formalize them in models, relate them with other concepts, analyze them to solve a series of information systems engineering issues such as, several scenario-based techniques [Rolland 1998b]. However many questions still remain: are all intentions identified in this kind of approaches? Do the *theoretical* intentions in models fit with the actual real life intentions, and how to check this? Where do intentions come from if not from scenario analysis? As process mining approaches aim at discovering and modeling activity process models from event logs, in the same manner, intentions could be identified and modeled using event logs. To the best of our knowledge, event logs have been neglected to model intentional process models so far. Therefore, the main challenge of this thesis is *how to identify and formalize intentions from event logs?*

1.3 Research Questions and Hypothesis

The lack of an automatic method to identify and formalize user's intentions from event logs motivates the contribution of this thesis. This thesis aims at developing a semi-automatic framework for the construction of intentional process models from event logs. To realize this, the following questions are addressed:

- Q_1 How can dependencies within activities be discovered to estimate the strategies? Event logs reflect which activities are really performed by users while enacting a process. The dependencies within activities can express different ways of process enactment.
  - $H_{1.1}$ It is possible to discover the dependencies within activities from event logs.
  - $H_{1.2}$ A strategy consists in a set of activities.
- Q_2 How can the intentions can be identified from the estimated strategies? Once the strategies are estimated, the link between these strategies and intentions must be discovered.
  - H_2 A strategy is followed to fulfill a given intention.
- Q_3 How can the processes be modeled at different levels of abstraction? The nature of granularity that is needed to model a process can be defined regarding the situation at hand.
  - H_3 Clustering techniques can be used to abstract the discovered intentions.

- Q_4 How can process mining techniques be adapted to automate process discovery to support variability and flexibility taking into account the users' intentions and strategies? Process mining enables the design of process from event logs resulting from information systems.
  - $H_{4.1}$ Process mining approaches only consider activity-oriented process models.
  - $H_{4.2}$ Process mining algorithms can be adapted to discover intention-oriented process models.

1.4 Research Method

The research method followed in this thesis consists of allowed making four phases:

1. Problem identification: as a first step of the research process, a systematic analysis of the existing literature was established to create the theoretical background of the current thesis. This allowed making a firm foundations of the research topic and the research methodology and finally outlined what are the contributions of this research to the existing body of knowledge. These sequential steps that were followed to find lacks within the literature consist of: searching the relevant literature, understanding the literature, analyzing the literature, synthesizing the literature, and evaluating the literature. To do so, among the more recent publications the peer-reviewed and non peer-reviewed publications were scanned in the digital library such as ACM, Elsevier, Springer, IEEE, in Google Scholar search engine, and many more. The scanning of the relevant literature allowed the identification of the problem. This revealed the lack of flexibility in actual mined process models, while empirical research shows integrating the flexibility in the processes is mandatory to maintain their alignment with the organizations' goals. During the problem statement phase, the main problem was fragmented into sub-problems that could be solved separately.

2. Determination of objectives: the main objective of this thesis focuses on the discovery of the intentions from event logs. This gives rise to the above-mentioned research questions. Therefore, the main objective can be divided into four sub-objectives. The first is obtaining different ways (strategies) to achieve the intention from event logs. The second consists in obtaining the intention from the obtained strategies. The third is automating intentional process discovery from event logs. The fourth is obtaining intentions in different levels of abstraction.

3. Design and development: the first research sub-question was addressed by adapting existing mining technique to the concept of the intentional process model. The second research sub-question was addressed by developing the new algorithms to discover intentions in different level of abstraction. All

the proposed solutions are developed in a tool. This tool allows obtaining intentional process models from event logs.

4. Demonstration and Evaluation: to demonstrate the proposed approach in practice, it is applied first on a example and then on a case study. This provides a systematic and rigorous manner of analyzing event logs, and reporting the results. This allowed investigating several phenomena in depth. According to [Yin 2009] there are several types of case study: exploratory, explanatory, and descriptive. The chosen case study for this thesis is descriptive and explanatory. The case study allows illustrating events and their specific context and linking an event with its effects and it is suitable for investigating causality among events. Regarding the objectives established in advance the generated outputs of the proposed approach are evaluated by a rigorous approach through well-known statistical metrics. The results of case study are then presented to the Eclipse developers to have feedback on their perception about the results. To do so, a detailed questionnaire is established which contains the questions about the developers' work habits, the obtained Map comprehension, etc.

1.5 Contributions of this Thesis

This thesis proposes a novel approach for process mining, called Map Miner Method (MMM) that mines users' activities within a given process using users' trace. A trace is considered as a set of events ordered by their timestamps and grouped by users. MMM stands out from process mining approaches by modeling the process in terms of users' intentions and strategies instead of users' activities. MMM takes users' traces as input to find users' intentions and strategies and constructs semi-automatically a Map process model [Rolland 1999]. More precisely, MMM consists in inferring the implicit users' intentions and strategies from users' traces recorded during the enactment of a given process. The discovery of intentions and strategies allows constructing a Map process model and thereby rebuilding the actual process model, *i.e.*, the model followed by users.

MMM is a generic approach, which can be useful at different stages of process model life-cycle, for instance: (i) at the requirements level, to semi-automatically construct the actual processes enacted by users in organization, from their daily traces recorded in traces; and (ii) at the project management level, to check the alignment between a prescribed process model and what users actually do, (iii) and possibly adapt and improve them to actual practice, (iv) at the application level, to monitor users and provide run-time recommendations.

MMM generates intentional process model specified with the Map formalism [Rolland 2005b]. In this thesis, this formalism is chosen rather than other goal-oriented formalisms such as i* [Yu 2011] or KAOS [Dardenne 1993] since, (a) it has already proven effective for specifying software engineering processes [Rolland 1993], (b) it supports process variability and multi-process specifi-

cation [Rolland 1999], (c) it combines intentions and strategies at multiple levels of abstraction [Rolland 2005b], and (d) it scales well to large and complex processes [Rolland 2009].

Discovering intentions from event logs is not limited to MMM. Mining humans' goals is a challenging issue that is widely studied today in different areas of research in computer science, engineering services, method engineering, security, sales, information retrieval, etc. This new discipline is called *Intention Mining*. The intention mining approaches have many applications for instance in the context of information retrieval [Outmazgin 2013, Yu 1987, Hashemi 2008, Baeza-Yates 2006, Chen 2002].

Intention mining approaches usually identify individual intentions for individual activities. Analyzing single activities leads to low-level intentions, also called basic intentions or action-intentions [Chen 2002], which are closer to activities than proper intentions. On the contrary, MMM offers a model that integrates the concept of high-level intentions by developing a new method that is based on extracting the information out of users' traces. This information (intentions and strategies of users) is then modeled by an intentional process model. Analyzing a sequence of activities allows determining the high-level intentions, *e.g.*, organizational goals. Indeed, intentions in a process are related to each other to reach the ultimate goal; thus intentions cannot be considered as independent entities resulting from a single activity.

From a technical point of view, MMM consists of three stages:

- Modeling users' activities: MMM uses Hidden Markov Models (HMMs) [Rabiner 1989] to model users' activities into a multi-level topology of users' activities and corresponding strategies. Then using different techniques of learning, the users' strategies can be estimated. Mining the intentions needs the design of new algorithms and tools to generate Map process models. In this perspective, two algorithms are developed:

- Deep Miner algorithm: from estimated users' strategies, Deep Miner algorithm identifies users' intentions and consequently it discovers Map process models with higher precisions and low-level intentions, which is called pseudo-Maps.

- Map Miner algorithm: starting from pseudo-Maps, Map Miner algorithm constructs Map process models by clustering low-level intentions into high-level intention with respect to the definition of intention in the Map formalism.

The entire proposed method was applied and validated on practical datasets, first in a laboratory context where a example was conducted with Master students of University of Paris 1 Panthéon-Sorbonne for Entity-Relationship diagrams creation. Second, MMM was applied, in a large-scale experiment, on event logs of developers of Eclipse UDC (Usage Data Collector) [Eclipse 2013] which demonstrates scalability of MMM. The resulting Map process models provide a precious understanding of the processes followed by the developers. A qualitative experiment was conducted

which allows evaluating of the case study. This evaluation provided the developers' feedback on the perception, effectiveness and usability of MMM in practical use. A tool, called Map Miner Tool, was also designed and developed to automate MMM. It enables practicing the proposed approach to obtain the personalized Map process model only from users' traces by adjusting some parameters.

1.6 Outline

The thesis is organized as follows:

- Chapter 2 provides a theoretical background of the related works in process mining and intention mining fields. At the end on the chapter a synthesis of existing approaches of these fields is given.
- Chapter 3 introduces the proposed approach and defines the position of the thesis regarding the literature.
- Chapter 4 describes in detail different parts of the proposed method. The entire of the method is then applied on a example.
- Chapter 5 presents the validation of the proposed approach by applying MMM on a large-scale case study.
- Chapter 6 describes the tool that is implemented from MMM.
- Chapter 7 concludes the thesis and expresses the perspectives of the thesis.

Chapter 2

State of the Art

Contents

This chapter gives a review of the literature for process mining and intention mining approaches categorized according to their typology, objectives, techniques, metamodels, open problems, and tools. At the end of the chapter a synthesis of these approaches presented.

2.1 Common Characteristics

Process mining and intention mining fields are two research domains related to this thesis. In this section several common characteristics of process mining and intention mining are investigated. This allows better understanding the divergence and convergence points of these fields. These characteristics are investigated according to several axes: input elements, users' sources, mathematical models, developed algorithms, ontologies, classification approaches, objectives, and output elements.

2.1.1 Input Elements

Some mainstream process modeling notations specify a process as a collection of activities such that the life-cycle of a single instance is described [Van der Aalst 2011c]. A process discovery is based on a fundamental element: event logs. Note that the assumption is that one event log corresponds to one process, which means only event logs relevant for the process under study are captured by the information systems.

Process mining and intention mining techniques aim at discovering process models out of event logs. Event logs are the starting point of both mining approaches. Event logs are generated by interactions of users with the information systems while enacting a process. The information systems logs provide accurate and valuable information about the underlying processes. Usually, an event log keeps information such as, the activities, the users, the timestamps, and the properties of the objects in use [Clauzel 2009]. Each event in process mining is assumed to be an activity carried out by a given user. An activity describes a well-structured step in a process.

Event logs provide useful information for diagnostics and auditing. They help to build theories of the user's cognition, to design better user interfaces, to have a precise model of the actual processes, to predict the user's behavior in specific conditions, to improve training techniques, or even to improve users' understanding of their activities [Georgeon 2012].

Event logs may have several quality levels according to their reliability, completeness and structure. Some possible levels are categorized as follows [Van der Aalst 2012]:

- First level: these kinds of event logs are of excellent quality, *i.e.*, reliable and complete and well-structured [Van der Aalst 2012]. In addition, event logs are recorded in an automatic, systematic (*i.e.*, an approach is followed to decide which events are recorded), reliable and secured manner. Security and privacy concerns are appropriately addressed. Moreover, these event logs have clear semantics, which implies the existence of one or more ontologies.

- Second level: these kinds of event logs are of less quality than those at the first level. They are recorded automatically and in a complete, systematic and reliable manner. Security and privacy concerns are moderately addressed.

- Third level: these kinds of event logs are recorded automatically, but not systematically. The event logs are trustworthy but not necessarily complete.
- Forth level: these kinds of event logs are also recorded automatically, but not systematically. Hence, some event logs are may be missing or not recorded correctly.
- Fifth level: these kinds of event logs are of poor quality. The event logs recorded by hand and may not correspond to reality and some events are may be missing.

Some authors also consider event logs as the users' activities enriched by the context of their enactment [Laflaquière 2006]. In this case, a *trace* is defined as a temporal sequence of observed items or, more precisely, a set of recorded data that are generated by users' interactions to complete their instrumented activities. A trace can be defined as a sign that an event has happened or existed [Cambridge 2013] or as a set of multiple streams of quantitative or symbolic data that record (at least partially) an activity performed by a user [Georgeon 2012]. There are ontology-based techniques that help to define transformation rules to process the raw traces into abstract traces [Georgeon 2012].

2.1.2 Users' Sources

Process mining and intention mining approaches can be applied for one user (individual) or many users (collective). The recorded logs can be mined for only one individual. In this case, the mined logs only represent the point of view of one user (obtained from his own logs). If logs are mined for many users, they will represent the point of view of a group of users or *crowd* (obtained from the logs of all the users).

2.1.3 Mathematical Models

Generally, the literature of process mining and intention mining suggests several sorts of mathematical models such as Bayesian Network (BN) [Friedman 1997], Dynamic Bayesian Network (DBN) [Murphy 2002], Markov chain [Martin 1967], Finite-State Machine (FSM) [Chow 1978], and Hidden Markov Models (HMMs) [Juang 1991].

A BN [Friedman 1997], also known as belief networks or Bayes nets for short, belongs to the family of probabilistic graphical models. These graphical models are used to represent knowledge about an uncertain phenomenon. A BN consists of several nodes and edges. Each node represents a random variable, and each edge between two nodes represents probabilistic dependencies among the corresponding random variables. These conditional dependencies in the graph are often estimated by using well-known statistical and computational methods [Jensen 1996a].

A DBN [Murphy 2002] is a BN, which models time series data and relates variables to each other over adjacent time steps. This is often called a two-time slice

BN because at any point in time t, the value of a variable can be calculated from the immediate prior value (time $t-1$). DBN simplifies the design of BN by assuming that an event can cause another event in the future and not in the past.

A stochastic process with the Markov property among a finite or countable number of possible states is named a Markov chain [Martin 1967]. A Markov chain should satisfy the Markov property - it should be a memoryless process, *i.e.*, the future state depends only on the current state and not on the previous one. In other words, the history of the process (all information of the previous states) is encapsulated in the current state. Another characteristic of a Markov chain is that the variables are discrete-time. The transitions are the changes of the system from one state to another and the related probabilities are called transitions probabilities. The transition matrix describes the transitions probabilities between all the states in a given process.

FSM [Chow 1978] is a mathematical model of computation used to design both computer programs and sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; which is called the current state. The state of the machine can change from one to another when an event or condition is triggered; this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition.

A HMM [Juang 1991] is a statistical signal modeling formalism that allows modeling a sequence by a finite number of states. HMMs are very flexible due to latent data that allows modeling the structure of complex temporal dependencies. The systems modeled by HMMs are based on two complementary Markov processes: hidden process and observed process. The hidden states are not observable, however they generate observations with different probabilities.

2.1.4 Techniques

Several techniques are used in process mining and intention mining approaches to discover intentions. These approaches may use ontologies, classification techniques, particular algorithms or a mixture of them. Hereafter, these techniques are briefly defined:

- Ontology-based: Some approaches use ontology. An ontology formally represents knowledge as a set of concepts within a domain, using a shared vocabulary to denote the types, properties and interrelationships of those concepts. Ontologies are the structural frameworks for organizing information and are used in artificial intelligence, the Semantic Web, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the world or some part of it. The main principles for constructing ontologies in Computer Science are found by [Gruber 1995]. These principles are such as classes, relations, functions, or other objects. Hereafter several definition are

given for an ontology: for instance, in Computer Science an ontology is defined as *"the capture of the recognized and conceived in a knowledge domain for the purpose of their representation and communication"* [Rebstock 2008]. An ontology is also considered as a novel and distinct method for scientific theory formation and validation [Akkermans 2006].

- Classification-based: Classification is the problem of identifying to which category a new observation belongs, on the basis of a training set of data containing observations whose category membership is known. In the terminology of machine learning, classification is considered as a type of supervised learning, *i.e.*, learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering (or cluster analysis), and involves grouping data into categories based on some measure of inherent similarity.

- Learning-based: some approaches use well-known algorithms such as Baum-Welch algorithm [Baum 1970]. Others developed and designed either their own algorithms or a mixture of well-known and particular algorithms to discover the pattern underlying logs.

2.1.5 Objectives

The process mining and intention mining approaches focus on discovery of information from logs. Process mining approaches aim at discovering activities from logs and represent them in process models. However, process mining approaches may tackle with conformance (*i.e.*, verifying the gap between the prescribed model and the discovered model), enhancement (*i.e.*, improving the prescribed model) or recommendation (*i.e.*, providing the recommendations to users). Intention mining existing approaches mainly focus on intention discovery from logs. Nevertheless, a few propose recommendation techniques from discovered users' intentions.

2.1.6 Output Elements

The process mining approaches may generate either activity-oriented process models (instance of a defined process metamodel). It is then important to highlight the metamodeling concept. A model that represents a modeling language is called a metamodel. The model is an instantiation based on the metamodel. The relationship between a model and metamodel is called an instance of relationship [Group 2013]. Existing intention mining approaches allow discovering a single output: an individual intention.

Next section investigates the process mining field under different aspects.

2.2 Process mining

Fueled by the growing presence of event logs in software and software engineering platforms, process mining idea has initially emerged in the software engineering field with Cook and Wolf [Cook 1998a] and then has been applied on workflow log for the first time by Agrawal [Agrawal 1998]. Process mining gathers information about the processes as they are actually enacted, by making the assumption that it is possible to record events along with the order in which they are executed [Van der Aalst 2004c]. Indeed, retrieving event logs containing information about the actual process allows having an insight into the followed process model [Van der Aalst 2011c]. Process mining aims to fill the gap between activity traces obtained from event logs and process models.

Van der Aalst defines *"the idea of process mining is to discover, monitor and improve real processes, (i.e. not assumed processes) by extracting knowledge from event logs readily available in today's systems"*. He defines process mining as a bridge between data mining and process modeling and analysis [Van der Aalst 2011c].

Process mining uses data mining techniques to mine data logs containing process enactment data to reconstruct actual business processes [Tiwari 2008]. Data mining techniques allow extracting knowledge from large data sets through identification of patterns within the data. They were developed and adapted to create the process mining techniques. Furthermore, several customized algorithms have also been developed specifically to address the needs of process mining [Tiwari 2008].

2.2.1 Typology

The term *process* is defined as *"a set of partially ordered steps intended to reach a goal"* [Feiler 1993]. It is also defined as a collection of related, structured activities or tasks that produces a specific service or product (*i.e.*, a manner to reach the goal determined by products) [Olle 1988]. The process allows knowing: *"what really happened in the past, why did it happen, what is most likely to happen in the future, when and why do organizations and people deviate, how to control a process better, how to redesign a process to improve its performance"* [Rozinat 2010].

A process is expressed in terms of a process model. According to Rolland [Rolland 2005a], *"a process model prescribes a way to make a methodological approach to achieve the desired target. It describes an abstract level and ideally how to organize the production of the product: steps, activities along with their scheduling, and sometimes the criteria for moving from one stage to another. It acts as a mold for engineering process"*. Process models describe the common characteristics of a category of processes having the same nature [Rolland 1998a].

The first classification of process models was made by Dowson [Dowson 1987]. It comprises three types of process models: activity-oriented, product-oriented and decision-oriented. Later on, two other categories have been added to this topology: context-oriented [Rolland 1994] and strategy-oriented [Rolland 1999].

The next sections present the different types of existing process in chronological

order. Each type represents a process notation.

2.2.1.1 Activity-oriented Process Notations

Activity-oriented process models concentrate on the activities and tasks performed in producing artifacts and their ordering. They represent the activities and their scheduling for the realization of a product [Rolland 2005a].

Early models of activity-oriented process models appeared in the 70s with linear models such as, the V model [MacDermid 1984], the Cascade model [Royce 1970], which have evolved toward iterative models such as, the Fontaine model [Henderson-Sellers 1990], and also the Spiral model [Boehm 1988]. Thereafter, the iterative and incremental process models were introduced with Rapid Application Development (RAD) [Martin 1991] then with unified processes such as, the Rational Unified Process (RUP) [Kruchten 2004], 2 Track Unified Process (2TUP) [Roques 2004], and Symphony [Hassine 2002]. The processes using agile methods are also activity-oriented such as, SCRUM [Schwaber 2002], and XP [Beck 2001].

2.2.1.2 Product-oriented Process Notations

Product-oriented process models are similar to activity-oriented process models. They link the products states to the activities that generat these states. They focus on products rather than activities. They are comparable to state-transition diagrams [Rolland 2005a]. Indeed, the state of a product models its situation at a given instant of the process. Transitions are defined between these states to model the order in which the states may change. Transitions are relationships between a source state and a target state, which are triggered by an event. The product-oriented process models are created as required, when some products have specific and complex conditions that must be specified.

The template ViewPoints [Finkelstein 1991], the metamodel of Statecharts [Harel 1987], state machines in UML [Group 2013] as well as the metamodel of Entity Process Model (EPM) [Humphrey 1989] are some of the metamodels of product-oriented process models.

2.2.1.3 Decision-oriented Process Notations

Decision-oriented process models introduce the concept of reasoning, choice and decision-making, the processes are then seen as teleological. Decision-oriented process models represent transformations of products or successive elicitations due to decisions [Rolland 2005a].

In information systems engineering context, it is important to know why an activity was performed rather than another. Moreover, it is important to keep track of interactions between the various users. On the one hand, this allows better understanding the reasons for which the choices are made, and on the other hand, permits monitoring whether decisions taken during the information systems

engineering process are rational [Jarke 1992]. Regarding the problems complexity or situation feature in each project, different models of decision-oriented processes could be created.

The notion of decision-oriented process was introduced for the first time in Issue-Based Information Systems (IBIS) [Kunz 1970]. This model was implemented in a tool called gIBIS [1] to manage the discussions, the problems encountered during the upstream phases of the system design. It was then enriched by [Potts 1988]. The project European ESPRIT DAIDA has also developed a decision-oriented process model, called CAD [2].

2.2.1.4 Context-oriented Process Notations

Context-oriented process models are composed of a *situation* and an *intention* of a user at a given time of the project [Rolland 2005a].

A situation is any part of the product under development that may be subject to a decision [Plihon 1996]. Contextual process models relate the context of a decision to the decision itself [Rolland 1998a]. They allow changing the current situation to a new one by applying decisions to the situation.

The metamodel of context-oriented process NATURE was initialized by Rolland [Rolland 1994], then was extended by Plihon [Plihon 1996] and Rolland [Rolland 2000]. The PRIME environment [Pohl 1999] was inspired by the planning paradigm, which is used in Artificial Intelligence GRAPPLE [Huff 1987]. The concept of metamodel for context-oriented process models was extended for modeling strategy-oriented process models, which provide different ways to fulfill intentions.

2.2.1.5 Strategy-oriented Process Notations

Strategy-oriented process models (Map [Rolland 1999]) are an extension of context-oriented process models that intend to represent several levels of abstraction in the same process model. This multi-level of abstraction provides several possible ways to develop the product. It is based on the concepts of *intention* and engineering *strategy* to achieve these intentions [Rolland 2005a]. According to this process model an intention is defined as a goal that can be achieved by the performance of a process [Rolland 2007], and a strategy is defined as an approach, a manner or a mean to achieve an intention [Rolland 2007].

We consider that there is only one strategy-oriented process model, which is called CREWS-L'Écritoire [Ralyté 1999]. It allows discovering and identifying requirements in information systems context. Strategy-oriented process models are also used in other cases. For instance, they are used to model a method for similarity analysis between the business requirements of an organization and the functional

[1] graphical IBIS [Conklin 1989]
[2] Conversation among Agents on Decisions over objects [Jarke 1992, Rose 1991]

requirements of an information system [Zoukar 2005]. They are used to model the alignment between business processes and information systems [Etien 2006].

2.2.1.6 Synthesis on the Different Process Notations

This section discusses some advantages and drawbacks of the process notations described in the previous section with a process mining perspective.

Activity-oriented process notations have as the advantage [Arbaoui 1994] to provide precise and unambiguous process models. Indeed, in a given activity-oriented process all the activities as well as their order are prescribed and detailed in the model. This perfectly fits the structure of event logs. In other words, it is possible to extract activities, since they can be directly found in event logs.

However, activity-oriented process notations often fail in supporting real industrial process due to several drawbacks in modeling and performance support [Arbaoui 1994, Rolland 1998a]. Some of these drawbacks are described as follows:

- They are based on task or activity performance. This makes them rigid and inflexible. Indeed, process enactment is very restrictive and limited to pre-established model. They provide a frame for manual management of projects developed in a linear manner [Rolland 1998a]. They are most often treated as linear sequences where crucial aspects of the process such as feedback loops and iteration are not represented [Boehm 1988, Curtis 1988, Curtis 1992]. The linear view of activity decomposition seems inappropriate to model mined event logs of creative processes, since it does not permit considering all eventualities and consequences [Rolland 1998a].

- They overlook underlying users' intentions and do not allow reflecting about different choices made during the process enactment, since the information about users' intentions are not directly observable from event logs. Therefore, it is not possible to predict users' intentions in the future or to reason about their intentions and the different ways that users have selected to achieve their intentions.

- They require that all particular situations that may occur during the process enactment should be found in event logs. However, this information is not usually available in event logs. In this case, the mined processes do not reflect the actual ones.

- They emphasize only on activity, which can omit the impact of product structure on the mined process.

- They are inappropriate for modeling the mined event logs of processes that have to support parallel engineering, backtracking and reuse of previous conceptions, due to their linear vision of process models [Rolland 1998a].

Product-oriented process models, in a similar manner to activity-oriented ones, are centered around the notion of product state and link the product states to the activities that generate these states. They have several advantages: they are useful for tracing the transformations that occurred and their resulting products, they are also useful for tracing the transition between states of the products [Rolland 1998a]. However, since they adopt the notion of activity in their structure they have the same limitations and obstacles as the activity-oriented process models. Further, it is very difficult to mine product-oriented process models from event logs which are sufficiently realistic, since there is not sufficient information about the states and transitions of products in event logs.

The decision-oriented process models take into account more aspects of a process than product-oriented and activity-oriented process models, since they take into account the reasons *why* decision are made and *how* they trigger activities [Plihon 1996]. Decision-oriented process models have several characteristics [Rolland 1998a] such as:

- Supporting strategy-oriented processes for explanatory tracing and prescriptive guidance.
- Supporting the rationale underlying the decision process.
- Assisting in reasoning about the rationale underlying decisions.
- Directing the decision-making process.
- Preserving a history of the events that occurred in a process along with their reasoning.

Similarly to product-oriented process models, mining decision-oriented process models is not trivial, since it is necessary to have the information about decisions made by users in event logs. However, this kind of information is not usually available in event logs.

The decision-oriented process models have been completed for taking into account the situation in which a decision is made. These new process models are so-called context-oriented process models [Plihon 1996, Souveyet 2006]. The context-oriented process models adopt the notion of decision; thus they have all the properties (advantages and disadvantages) of decision-oriented process models. In addition, due to the strong correlations among the couple situation-decision, context-oriented process models take into account the situation in which adequate decisions are made. This allows tracing, providing guidance and explaining specific process situations. However, from a mining point of view, event logs usually contain basic information about the context of a process such as timestamps, date, etc. Therefore, mining such processes becomes more difficult.

In the strategy-oriented process models, the notion of intention has a fundamental place; thus, they can also be called *intention-oriented process models*. The main focus of such process models is on *what* the process is intended to achieve,

which implies providing a justification for *why* the process is enacted. The concept of *strategy* only belongs to this type of process models. The key idea of intention-oriented process models is to pursue the users' intention during the fulfillment of a goal as a force that guides the process. As a consequence, goals to be achieved are explicitly represented in the process model with the different alternative ways to achieve them, *i.e.*, strategies [Soffer 2005]. These strategies are composed of one or several activities. This property allows mining such process models from event logs. In other words, event logs naturally contain activities. These activities can be mined to find the strategies by using an appropriate technique. Once the strategies are found the intentions can be then found, since according to intention-oriented process models, there is a strong link between intentions and strategies.

2.2.2 Process Mining Objectives

Process mining techniques excerpt knowledge from event logs commonly available in information systems. The extracted information is then represented under the activity-oriented process models. These techniques provide new means to discover, monitor, and enhance processes in various application domains. They are able to provide detailed information about the history of processes to support and enhance business processes in competitive and rapidly changing environments. Mining processes from logs can be useful for understanding how people really work, analyzing how actual processes differ from the prescribed ones, and thereby improve the quality of information systems.

Van der Aalst classifies process mining techniques into three categories [Van der Aalst 2011c]: *Discovery*, *Conformance* and *Enhancement*. Later on, *Recommendation* techniques have emerged in some works [Schonenberg 2008, Mulyar 2008]. Hereafter, the four objectives of process mining are defined briefly:

- Discovery: the discovery techniques aim at discovering process models by analyzing event logs. There is no *a priori* information about actual process models. For instance, event logs may be studied by α-algorithm that automatically transforms them into a Petri net model [Van der Aalst 2011a]. This represents the users' behaviors as recorded in the event logs.

- Conformance: the conformance checking techniques use an *a priori* model to check the degree of alignment between the actual process model, *i.e.*, the model followed by users, and the prescribed process model. These techniques can detect the deviations from the prescribed model [Van der Aalst 2005a].

- Enhancement: the enhancement techniques use information recorded in event logs to improve and enrich the prescribed process model using methods of repair and extension [De Medeiros 2007]. Repair has a mirror effect, which means it tries to reshape the model to better illustrate reality. Extension allows widening the process model with new aspect by cross-correlating it with event logs.

- Recommendation: some techniques push forward the study of processes by using event logs to predict which activity may follow a current activity. For instance, [Mobasher 2000] proposes recommendations based on URL traces. Schonenberg et al. propose and experiment an approach based on recommendations. This approach shows the more history of process is used, the better is the quality of the recommendation [Schonenberg 2008].

2.2.3 Process Mining Techniques

The idea to apply process mining was introduced by Agrawal [Agrawal 1998]. At the same time, Datta proposed to discover business process models [Datta 1998] by using their own process mining technique. Cook et al. investigated similar issues in the context of software engineering processes [Cook 1998a]. The majority of the process mining techniques focus on process models discovery based on observed event logs [Van der Aalst 2004a, Agrawal 1998, Cook 1998a, Datta 1998, van Dongen 2004b, Weijters 2003, Herbst 2000b]. However, the process mining techniques are not limited to only process models discovery, for instance, social networks and other organizational models can be discovered from event logs [Van der Aalst 2005d, Song 2008]. An overview of the early work in this domain is given in [Van der Aalst 2003]. The most important process mining techniques are described as follows:

- Inference methods: these methods infer process models with a tradeoff between results accuracy and noise [3] robustness. Cook compares in [Cook 1995] three inference algorithms of RNet [Das 1994b], Ktail [Biermann 1972] and Markov models [Baum 1966] for process discovery. The latter two are considered as the most promising approaches. RNet [Das 1994a] is a statistical approach that characterizes a state depending on the past behaviors. RNet generates a Deterministic Finite State Machine (DFSM), *i.e.*, each state has only one transition for each possible input. Although it is robust to noise it is very time-consuming in the training phase, the size of the net increases with the number of token types, and it requires to evaluate many parameters. Ktail [Biermann 1972] is an algorithmic approach that evaluates the current state depending on future behavior. The input is a set of sample strings and the output is a Finite State Machine. The complexity of the algorithm can be controlled when the number of states increases by DFSM. The disadvantage of this method is that it is not very robust to noise. Markov models is a hybrid approach (*i.e.*, a statistical and algorithmic approach), which looks at the neighboring past behavior to define the future state. It has a DFSM and it is robust to noise with a controllable complexity. Cook and Wolf in [Cook 1998b, Cook 1999] proposed some techniques for concurrency detection and a measure to quantify the variance between behaviors and process models. Some other works use a Markov Chain Monte

[3] The deviation of the observed value from the inferred value

Carlo [Gilks 1996] technique for the discovery of frequent episodes in event sequences [Mannila 1997, Mannila 2001]. Metrics such as periodicity, entropy, event type counts and causality are proposed to discover models from event streams without generating explicit process models.

- α-algorithm [Van der Aalst 2004a]: this algorithm was proposed by Van der Aalst et al. to rebuild the causality in the Petri nets workflow from the existing relations in the event log. α-algorithm takes the event logs as input, rebuilds process models by using simple XOR, AND splits and joins; thereby creates the workflow nets as output. α-algorithm cannot handle noise and certain complicated routing constructs of workflow nets such as, loops and long-term dependencies, particularly during complex situations [Rozinat 2010]. A more robust but less precise approach was then proposed to deal with the issues of α-algorithm [van Dongen 2004b]. To overcome this difficulty an extended algorithm, α ++ algorithm [Wen 2006], was introduced to generate new relationships between event logs to handle long-term or implicit dependencies.

- Directed acyclic graphs [Agrawal 1998]: this approach proposes to transform events into dependency graphs or workflow graphs using directed acyclic graph, representing events and their causal relations without loop. These workflow graphs are influenced by workflow management products such as, IBM MQSeries [IBM 1999] workflow and InConcert [Tibco 2000]. The approach of Agrawal deals with problems of finding a workflow graph, creating event logs, and defining the edge conditions. However, using this kind of graphs to model the processes is delicate due to the existing loops in process models. To overcome this issue, the work tries to count the tasks frequencies and then fold the graph. Nevertheless, the results are only partially satisfying as the model does not completely fit the actual process [Tiwari 2008].

- Inductive workflow acquisition [Herbst 1998, Herbst 2000a, Herbst 2000b, Herbst 2004a, Herbst 1999]: in the context of workflow management, this technique aims at finding the best Hidden Markov Models (HMMs) [Rabiner 1989] that reflect the process models acquisition out of workflow models as well as their adaptation to requirements changes. This consists of two steps: induction and transformation steps. In the induction step, it is possible to find the HMMs by merging or splitting models to discover the underlying process. Each state of HMMs corresponds to a task node. The event logs can be observed and generated into workflow nets by inductive learning. The approach, described in the aforementioned works, also uses stochastic task graphs to generate a workflow model (transformation step) expressed in the ADONIS modeling language. This approach supports occurrence of the same tasks several times in the model (duplicate tasks). It is similar to the approach of directed acyclic graphs due to the presence of the splits and joins in the transformation step. This approach allows for concurrency. A notable difference of this technique with others is that the same task

can appear multiple times in the workflow model, *i.e.*, the approach allows for duplicate tasks. However, some works using this approach are limited to sequential models [Herbst 1998, Herbst 1999, Herbst 2000b].

- Hierarchical clustering [Greco 2005b, Song 2009]: this technique separates a set of event logs for a given process into clusters and finds the dependency graph for each log. It structures the clusters of event logs into a hierarchy tree. For each cluster, a workflow model is constructed and finally all the models are merged into a single one. Some clustering techniques use theory of regions to discover processes [Van der Aalst 2010, Van der Werf 2008, Carmona 2008]. The advantage of the theory of regions is that the characteristics of the resulting model can be influenced before the mining starts (*e.g.*, the number of places in the Petri net or the number duplicate task can be determined in advance). A mining tool has been developed for discovering hierarchically structured workflow processes that need to balance splits and joins [Schimm 2003, Schimm 2004].

- Genetic algorithm [De Medeiros 2005b]: this technique provides process models (Petri nets) built on causal matrix, *i.e.*, input and output dependencies for each activity. This technique tackles problems such as, noise, incomplete data, non-free-choice constructs [4], hidden activities, concurrency, and duplicate activities. Nevertheless, it requires the configuration of many parameters to deal with irrelevant data, which is a complex task.

- Heuristic algorithm [Weijters 2003, Van der Aalst 2002b, Weijters 2001]: this technique is based on α-algorithm. It calculates the frequencies of relations between the tasks, *e.g.*, causal dependency, loops, etc, and construct dependency/frequency tables and dependency/frequency graphs. This technique can detect irrelevant logs. However, like the Genetic algorithm, Heuristic algorithm needs a complex configuration phase. More recently, to deal with less structured, *i.e.*, very diverse or flexible processes, dynamically adaptive process simplification algorithms have been proposed [Günther 2007]. The approach [Van der Aalst 2002b] demonstrates that for some subclasses, it is possible to discover the accurate workflow model using α-algorithm. In another work, an extended version of α-algorithm is used to include the timing information [Van der Aalst 2002a].

2.2.4 Metamodels Used in Process Mining

There are several metamodels for representing activity-oriented process models such as, declarative process models [Pesic 2006], Event-driven Process Chains (EPCs) [Van der Aalst 1999], Petri Nets [Peterson 1981], Business Process Model and Notation (BPMN) [Object Management Group 2013].

[4] Free-choice Petri nets are Petri nets in which there are not two transitions consuming from the same input place but there is an input place, which is not an input place of the other [Desel 1995]

Declarative models specify what should be done without specifying how it should be done [Pesic 2006]. This approach proposes the ConDec language for modeling and enacting dynamic business processes. ConDec is based on temporal logic rather than some imperative process modeling language. In declarative process models approach a fundamental paradigm for flexible process management are proposed.

EPCs are the ordered graphs of events and functions providing various connectors that allow alternative and parallel execution of processes. Furthermore, they are specified by the usages of logical operators, such as OR, AND, and XOR. According to [Tsai 2006], *"a major strength of EPCs is claimed to be its simplicity and easy-to-understand notation. This makes EPCs widely acceptable technique to denote business processes."*

The most used metamodels in process mining are Petri nets and BPMN, which are described hereafter:

Petri Nets. Petri nets [Peterson 1981] are mathematical modeling languages allowing to model concurrency and synchronization in distributed systems. They are used as a visual communication aid to model the system behaviors [Van der Aalst 2011a] and to represent the process mining results. A Petri net is a directed graph composed of three types of components: places, transitions, and arcs. Each place represents a possible system state; when events or activities occur, transitions allow going from a place to another. Arcs maintain the relations between transitions and places.

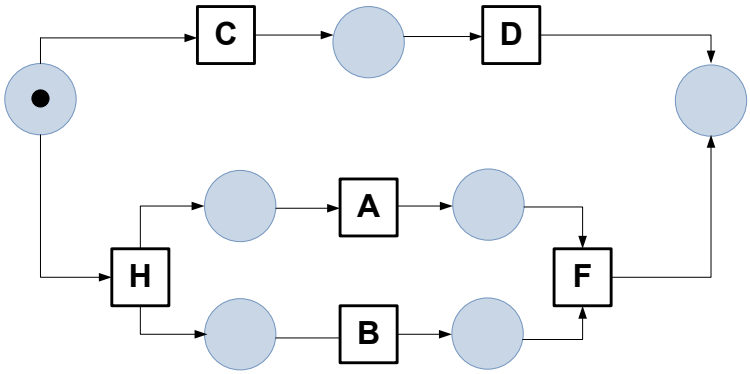

Figure 2.1: A process modeled with a Petri net [Van der Aalst 2004c]

Figure 2.1 illustrates a process model by a Petri net. The sequence of activities (places) are shown by **A**, **B**, **C**, **D**, **F**, and **H**.

Business Process Model and Notation (BPMN). BPMN [White 2004] is a graphical diagram to model business processes; it aims at providing an easy graphical way to model business procedures that are understandable by all business users. Furthermore, one can model complex business process readily by map it to other languages such as, BPML (Business Process Modeling Language), BPEL4WS

(Business Process Execution Language for Web Services) or UML. Figure 2.2 depicts a process model by BPMN.

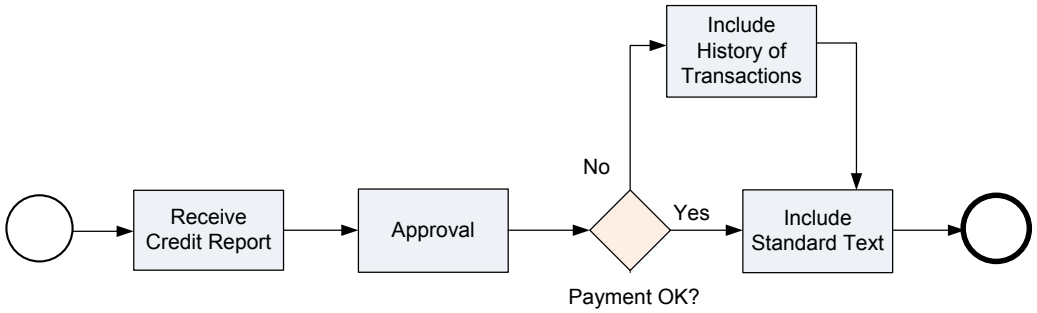

Figure 2.2: The process model represented by a BPMN

BPMN creates a standardized link to fill the gap between business process modeling and implementation procedures. It improves the possibilities of traditional notations by managing the complex nature of internal and business-to-business processes interactions by providing a standard notation readily understandable by all business stakeholders.

2.2.5 Process Mining Open Issues

Process mining still encountered a number of issues [Van der Aalst 2004c]. Whereas, several works attempt to address many of these problems, others require extra research to be solved. The main issues still unresolved are outlined as follows:

- Noise: it is possible that the event logs contain *noise*. This could be incomplete, incorrect logged information, unrecorded event, etc. The process mining techniques should be robust to noise, for instance by determining a threshold value to avoid exceptional or incorrectly logged behavior. Several works try to deal with noise [Weijters 2001, Maruster 2001, Weijters 2003, Cook 1998a, de Medeiros 2004, Gaaloul 2005, van Dongen 2004b, Agrawal 1998, De Medeiros 2005b, Cook 1998a].

- Hidden tasks: process mining techniques can only find information about tasks for which related event logs are recorded. However, it is possible to have *hidden task* in a process, which are not recorded in data logs. Finding hidden task is more difficult for more complicated processes. This issue is related to unobservable behaviors and branching or bisimulation issues [Van Glabbeek 1996]. Some works try to deal with hidden tasks [De Medeiros 2005b, Van der Aalst 2005b, Van der Aalst 2005c]

  However, in very simple process it is possible to find hidden task. For instance, let us consider a process with tasks **A**, **B**, **C**, **D**, **F**, **H**. Figure 2.1

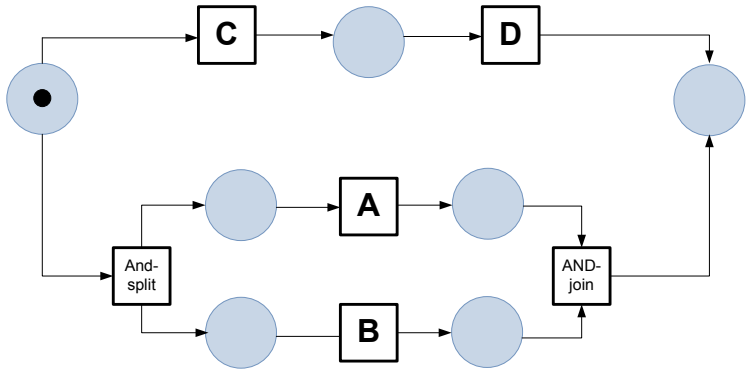

Figure 2.3: A process with two hidden tasks [Van der Aalst 2004c]

depicts this process modeled with a Petri net. Now, let us consider a case in which all event logs referring to task **H** are eliminated from recorded data [Van der Aalst 2004c]. The new discovered process is shown in Figure 2.3. In this particular case, it is still possible to model a similar process. As found on the figure, tasks **A** and **B** are executed in parallel, which means there has to be an AND-split. Moreover, for the same reason, if all the event logs referring to task **F** are also eliminated from recorded data, there has to be an AND-join.

- Duplicate tasks: this problem happens when two process nodes may refer to the same process model, which means an occurrence task in a given case [Van der Aalst 2004c]. Duplicate tasks are related to hidden tasks in the sense that a given process with hidden tasks but with no duplicate tasks can be modified into equivalent process with duplicate tasks but with no hidden tasks [Van der Aalst 2004c].

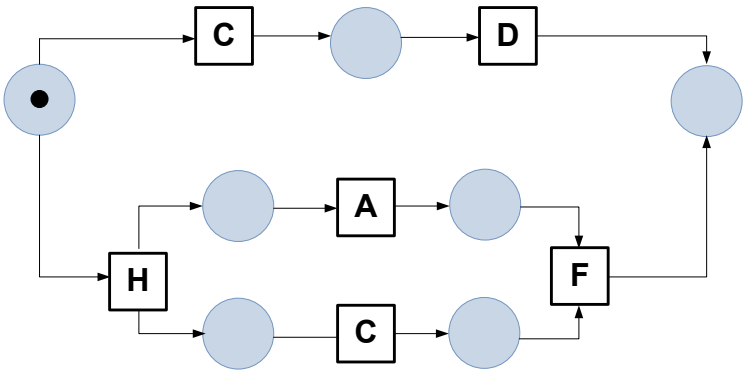

Figure 2.4: A process with duplicate tasks [Van der Aalst 2004c]

An example of duplicate task is shown in Figure 2.4. In this case task **B** is renamed to **C**. Obviously, it is not possible to differentiate task

B from **C**. Several works propose some solutions to overcome the duplicate tasks problem [Herbst 2004b, Van der Aalst 2005c, Van der Aalst 2005b, De Medeiros 2005b, Agrawal 1998].

- Non-free choice constructs: this problem happens in Petri nets where there are no two transitions consuming from the same input place but where one has an input place which is not an input place of the other [Desel 1995]. They are controlled choices that depend on choices made in other part of the process model. This excludes the possibility to merge choice and synchronization into one construct. Non-free-choice constructs are difficult to model. For instance, Figure 2.1 depicts a free-choice Petri net, since synchronization (task **F**) is separated from the choice between **F** and **C**. Figure 2.5 shows a non-free-choice construct. After executing task **B** there is a choice between task **F** and task **C**. However, the choice between **F** and **C** is controlled by the earlier choice between **H** and **A**. Note that tasks **F** and **C** are involved in a choice but also synchronize two flows. Clearly such constructs are difficult to mine since the choice is non-local and the mining algorithm has to remember earlier events [Van der Aalst 2004c]. Some research works tackle the non-free-choice issue [Van der Aalst 2002b, De Medeiros 2005b, Van der Aalst 2005b].

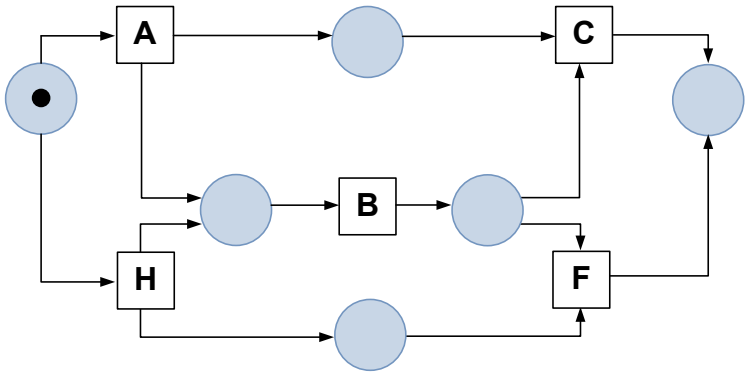

Figure 2.5: A process model with a non-free-choice construct [Van der Aalst 2004c]

- Mining loops: a process may be executed multiple times, this case refers to a loop, which may be simple or complex, *i.e.* involving one or more events. In the case of complex loops, mining is not a trivial task since, there are multiple occurrences of the same task in a given instance of process. Let us consider a simple example with a loop depicted in Figure 2.6. As found in this figure, task **A** can be executed several times after task **H**.

  In other words, the possible sequences are **HAF**, **HAAF**, **HAAAF**, **HAAAAAF**, and so forth. However, in complex processes, mining loops is a challenging issue particularly for the cases with backward loops. Several works attempt to overcome or to mitigate the problem of mining loops [Weijters 2003, de Medeiros 2005c, de Medeiros 2004,

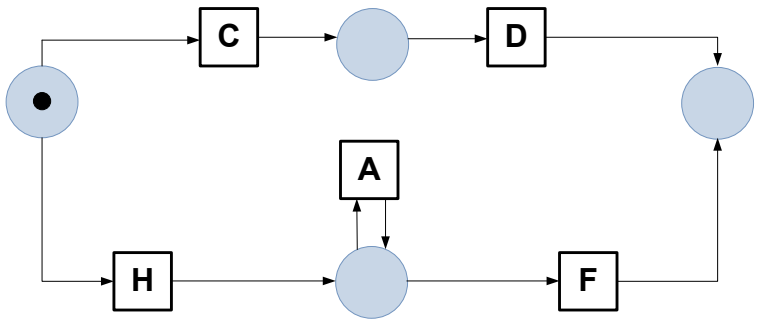

Figure 2.6: A process with a loop [Van der Aalst 2004c]

Medeiros 2005, Herbst 2004b, Van der Aalst 2004b, Van der Aalst 2005c, Van der Aalst 2005b, Weijters 2001, Schimm 2003].

- Incompleteness: this problem, related to the noise issue, occurs when event logs contain insufficient information to drift the process. For instance, when some activities are missing in recorded logs. To tackle this problem heuristics are used. These heuristics are typically based on Occam's Razor or the Minimum Description Length (MDL) principle [Rissanen 1978].

- Different perspectives: process mining mainly based on the control flow perspective fueled by the ordering of timestamped-tasks. However, further information may be append to the events such as, the organization perspective, the information perspective, and the application perspective. For instance, the application perspective deals with the applications being used to perform tasks. Therefore, it is challenging to mine the processes with different perspectives [Van der Aalst 2004b].

- Delta analysis: The results of process mining are the process models. Therefore, it is interesting to compare the prescribed models with the models resulting from process mining. Delta analysis is used to compare the prescribed process models with the discovered process models and then to check similarity or disparity between them. There are few techniques of delta analysis, which use behavioral inheritance [Van der Aalst 2001] or quantitatively measuring [Cook 1999] to discover differences and commonalities of process models [Van der Aalst 2005a].

- Visualizing results: the results of process mining approaches should be presented in a graphical way to make easier the understanding of process. However, visualizing the complete control flow perspective or other perspectives is more difficult and requires further research. ARIS PPM [Fischer 2008] is a commercial tool, which enables visualizing the complete control flow and allows focusing on the flow time

and work in progress to measure and to analyze the performance and structure of business processes. Several other works in this domain are [van Dongen 2005c, Cook 1998a, de Medeiros 2005c, van Dongen 2004a, Herbst 2004b, Dustdar 2005, Van der Aalst 2005b, Hammori 2004].

- Heterogeneous results: the required information for process mining approaches is dispersed over information systems. Therefore, the event logs, which are input of process mining, are not readily accessible. This problem is more complex when information systems are based on different platforms [van Dongen 2005b]. One approach is to use a data warehouse which extract the information from these logs [Eder 2002].

- Concurrent processes: sometimes processes occur at the same time. Mining such processes is then challenging. Some works focus on concurrent behaviors of processes [Cook 1998b, Herbst 2000a, Weijters 2001, Cook 1998a, Schimm 2004, Van der Aalst 2005b, Cook 1998b, Golani 2003, Zhang 2003, de Medeiros 2005c, de Medeiros 2004]. They present some techniques to discover patterns of concurrent behavior out of workflow traces. The techniques are based on a probabilistic analysis of the traces.

- Local/Global dimension: Mining algorithms aims at discovering the most appropriate process model among the potential candidate process models. There are different strategies to discover the most suitable process model: local and global strategies. Local strategies build step by step the optimal process model using local information. Global strategies build the optimal process model by one strike search. Some works try to deal with this issue [Greco 2004, Van der Aalst 2005b].

Tables 2.1 and 2.2 indicate the process mining issues and the research works dealing with them.

Research Works	Mining Loops	Hidden tasks	Delta Analysis	Non-free Choice Constructs	Visualizing Results	Heterogeneous Data Source	Local/Global Search	Process Rediscovery	Different Perspectives	Duplicate Tasks	Noise	Concurrent Processes
[Van der Aalst 2005a]			×									
[Greco 2004]							×					
[Cook 1998b]												×
[Van der Aalst 2002b]				×				×				
[van Dongen 2005c]					×							
[Weijters 2003]	×							×			×	
[Golani 2003]												×
[Cook 1998a]					×						×	
[de Medeiros 2005c]	×				×							
[Zhang 2003]												×
[de Medeiros 2004]	×										×	
[Gaaloul 2005]											×	×
[van Dongen 2004b]											×	
[van Dongen 2004a]					×							
[Medeiros 2005]	×											
[van Dongen 2005a]										×		
[Agrawal 1998]										×	×	
[De Medeiros 2005b]		×		×							×	

Table 2.1: Research works dealing with process mining issues

Research Works	[Cook 1998b]	[Herbst 2004b]	[Dustdar 2005]	[Van der Aalst 2004b]	[Hwang 2004]	[Van der Aalst 2005c]	[Van der Aalst 2005b]	[Schimm 2004]	[van Dongen 2005b]	[Hammori 2004]	[Schimm 2003]	[Weijters 2001]	[Cook 2004]
Mining Loops		×		×		×	×				×	×	
Hidden tasks						×	×						
Delta Analysis													
Non-free Choice Constructs							×						
Visualizing Results		×	×				×			×			
Heterogeneous Data Source									×				
Local/Global Search							×						
Process Rediscovery													
Different Perspectives				×									
Duplicate Tasks		×				×	×						
Noise					×	×	×					×	×
Concurrent Processes	×						×	×				×	×

Table 2.2: Research works dealing with process mining issues (following)

2.2.6 Process Mining Tools

Several tools exist to support the process mining techniques such as, ProM [Van der Aalst 2009], CPN tools [Jensen 2007], EMiT [van Dongen 2004a], Disco tool [Disco 2014], InWoLvE [Herbst 2004b], etc.

The ProM framework is a pluggable framework that supports various plug-ins for different techniques of process mining such as, α- algorithm and its extensions. This framework encourages a unified approach to be adopted in the design of process mining software. The framework is flexible with respect to the input ad output format, and it allows reusing code during implementation of new concepts [Van der Aalst 2009]. ProM has a generic interface capable of hosting a range of process mining and modeling activities.

CPN tools allow modeling and analyzing CPN models [Jensen 1996b] and simulating processes to analyze and check them. A combination of CPN tools and ProM framework plug-in is implemented to improve business processes modeling [De Medeiros 2005a, Rozinat 2008a]. Disco tool allows mapping automatically the event logs with CSV and XLS extensions to the appropriate XES or MXML notations, which are supported by ProM. This permits having an insight into the processes from event logs very quickly. It optimizes performance, controls deviations and explores variations. However, it is a commercial tool and does not provide information about the used algorithms. EMiT is able to integrate timing information using an extended version of α-algorithm. This tool transforms the event logs of commercial systems to XML format. It tries to find the causal relations among logs and based on that, rebuilds a Petri net represented in a graphical model.

InWoLvE workflow is a mining tool, which deals with four classes of workflow models [Herbst 2004b]: injective sequential, injective parallel, non-injective sequential and non-injective parallel. The term injective relates to the uniqueness of the activity nodes of the process models. An injective problem class means that each node of the process is unique (there are no repetitions of nodes in the process model). A sequential problem class (process model) does not contain any splits of joins, unlike a parallel problem class. The ability of this tool to mine process models with non-unique nodes is an advantage.

Some tool such as, ARIS [Scheer 2014], PISA tool [Zur Muehlen 2000], and SPM [Staffware 2014] do not extract the process model, but they aim at clustering and analyzing the performance rather than causal relations. ARIS [Scheer 2014] tool designed to manage the process performance. Staffware Process Monitor (SPM) [Staffware 2014] is customized to mining Staffware logs. PISA tool [Zur Muehlen 2000] can extract performance metrics from workflow logs.

Next section investigates intention mining field according to different aspects.

2.3 Intention Mining

Mining humans' intention is a challenging issue in a wide range of domains. They define intention in different ways, use them for various objectives, apply different techniques to discover intentions, etc. This give rise to a new discipline of research, so-called **Intention Mining**.

Intention is widely used in different areas of research in computer science: engineering services [Rolland 2010], method engineering [Rolland 1993, Deneckère 2010, Salinesi 2003, Mirbel 2006, Najar 2011], for context adaptation [Ralyté 2003, Mirbel 2006], business process [Outmazgin 2013], requirements definition [Yu 1987, Van Lamsweerde 2001], information retrieval [Hashemi 2008, Baeza-Yates 2006, Chen 2002].

In the broad sense, intention mining refers to all approaches that aim at discovering intentions from interactions of users with information systems. This refers to uncover users' intentions from users' activities (*e.g.*, logs, traces, etc).

Next section investigates a review of intention mining approaches in different aspects.

2.3.1 Typology

The notion of *intention* is a common word in an everyday context. From a psychological point of view, intention is defined as: *"our common sense psychological scheme admits of intentions as states of mind; and it also allows us to characterize actions as done intentionally, or with a certain intention"* [Bratman 1999]. Another definition considers an intention as: *"a determination to act in a certain way; a concept considered as the product of attention directed to an object or knowledge [· · ·]"* [Gove 1981].

Intentions are defined as the stable *goals* and of higher-order entities that function as abstract, organizing structures and remain fairly stable over time [Chulef 2001].

Beside of the humans' cognitive context, the *intention* is a manifold notion in a wide rang of domains. Hereafter, the notion of intention is presented from points of view of some salient computer science disciplines such as information systems, requirements engineering, and information retrieval.

2.3.1.1 Intention in Information Systems Context

In information systems context, the concept of purpose, goal or intention is essential for any organization since, information systems are created to fulfill the organization needs. An intention is considered as a property of an active component that has choice of behavior opposed to passive component [Feather 1987, Fickas 1992].

In the middle 90s, Jackson defines an intention as an optative statement, a state or a result expected to be reached or maintained in the future [Jackson 1995]. A goal is also defined as an objective the system under consideration should achieve;

thus goal formulations refer to intended properties to be ensured; they are bounded by the subject-matter expert [Zave 1997].

Intentions, also referred to as *goals*, *objectives*, or even *mission*, *purpose*, *intents* are a first class concept of information systems engineering [Rolland 2005b]. The notion of intention can also be seen as a goal that a user wants to achieve without saying it [Rolland 2007].

Some approaches do not explicitly deal with intention notion but with the similar concepts such as, *workarounds* [Outmazgin 2013], which are non-compliant behaviors. The notion of *workarounds* could be considered as intentions since, they express the desire of a user to perform a specific task. Business process workarounds situations are the circumstances that users are aware of the required internal procedures and intentionally decide to act differently. They can be observed when people deviate, in full knowledge, from the processes they are supposed to follow. The driving idea is that detecting and understanding workarounds make it possible to improve process models. The concept of workaround draws a clear link between process models and intentions. First, workaround patterns point out to implicit intentions. Second, the method relates these *actual* implicit intentions, *i.e.*, from peoples' behavior observations, with *theoretical* explicit intentions, *i.e.*, from process models.

2.3.1.2 Intentions in the Requirements Engineering Context

In the requirements engineering context, intention captures at different levels of abstraction, the various intentions the system under consideration such as, achieving requirements completeness [Yu 1987], avoiding irrelevant requirements [Yu 1987], explaining requirements to stakeholders [Mostow 1985, Lee 1991], structuring complex requirements documents, providing many alternatives for engineers, managing conflicts, separating stable from more volatile information, for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying requirements [Van Lamsweerde 2001].

In the requirements engineering goal is beneficial as it allows supporting heuristic, qualitative or formal reasoning schemes during requirements engineering [Van Lamsweerde 2001]. Goals may be formulated at different levels of abstraction: high-level such as, strategic concerns and low-level such as, technical concerns. Goal also cover different types of concerns: functional concerns such as, the services to be provided, and non-functional concerns such as, quality of service, *e.g.*, safety, security, accuracy, performance, etc). The system to which a goal refers is the current one or the system-to-be; both of them are involved in the requirements engineering process. High-level goals often refer to both systems. Two types of goal-oriented modeling are described in [Thevenet 2007a]: models of *operational goals* that describes goals considering business processes and systems functionalities, and models of *strategic intentions* that describe the organization strategy. A distinction is made between soft goals, which cannot satisfy clear-cut criteria [Mylopoulos 1992] and hard goals, which can be satisfied via inspection tech-

niques [Dardenne 1993]. Soft goals are especially useful for comparing alternative goal refinements and choosing one that contributes the best to them.

2.3.1.3 Intentions in Information Retrieval Context

In the information retrieval context, the key idea is better understanding the rationale behind the users' activities through Web engine. This can be useful to deal with a range of issues such as, recognizing users' intentions, reasoning about them, or generating plans to help users to achieve their intentions [Strohmaier 2012, Hashemi 2008, Baeza-Yates 2006, Park 2010, Jethava 2011, González-Caro 2011]. Most of the intention mining techniques focus on mining individual intentions out of Web engine queries. This research area is extremely dynamic with new contributions continuously published.

An intention is called *user intent* and presents a taxonomy of Web search according to the intent of the Web users [Broder 2002]:

- Navigational: *the intent is to reach a particular site*, when the users have an *a priori* information about the website such as the name of the website,
- Informational: *the intent is to acquire some information assumed to be present on one or more Web pages*,
- Transactional: *the intent is to perform some Web-mediated activity*, when the search leads to shopping websites or service-provider websites.

Later, the mentioned taxonomy has been used and adapted in the context of Web search [Baeza-Yates 2006]. This approach considers three categories:

- Informational: the intention is to acquire some kind of knowledge, information, data, etc, within the query Web,
- Not informational: the intention is to get other resources or to perform particular process, task, transaction, action, activity, etc.
- Ambiguous: the intention cannot be defined precisely in neither informational nor clearly in non-informational since, there is no clear-cut definition of it.

As explained in this approach, informational and non-informational intentions are not disconnected: achieving some non-informational intention, *e.g.*, cooking, may require to search for information, *e.g.*, a recipe, and the other way round, achieving a higher informational intention, *e.g.*, getting the list of ingredients, may involve some lower level non-informational intention, *e.g.*, converting weights from the imperial to the metric system.

The salient feature of Strohmaier and Kröll's approach is that, it differentiates between implicit and explicit intentions [Strohmaier 2012]. Implicit intentions underlie what is expressed by people or can be observed from them. Contrary to

explicit intentions, implicit intentions are neither expressed nor specified, nor indicated. They must therefore be labeled to become explicit, which is the purpose of this approach. For instance, they can be expressed in natural language for the predication of a sentence with an intention verb. As this work is applied to Web search query, a search query is regarded to contain an explicit goal whenever the query contains at least one verb and describes a plausible state of affairs that the user may want to achieve or avoid in a recognizable way.

An intention is also defined as a *concept considered as the product of attention directed to an object of knowledge* [Hashemi 2008]. This work aims at discovering users' intentions through the analysis of visited sites and distinguishes an intention as a single word (uni-token) or several words (multi-token).

Two sorts of intentions are defined as *action intentions* and *the semantic intentions* [Chen 2002]. The action intentions are low-level intentions, which are performed by basic actions on a computer such as, mouse click, keyboard typing, etc. The semantic intentions are high-level intentions and correspond to what the user wants to achieve at high-level, which may involve several basic actions on a computer to accomplish it.

The user's intention is also defined as personal and situational meanings that can satisfy a user's information need or goal in an exact way [Park 2010]. It consists of what to search, semantically-related word set, word set that represents a user's search target, personal/situational meaning that expresses a user's search need clearly and specifically. An intention map as defined in [Park 2010] is a set of situational meanings that can fully satisfy information needs of a user who uses queries with the same meaning. This map is inspired from the cognitive psychological knowledge representation method Schemata [Rousseau 2001]. It takes into account the intentions of the user, the query, and the corresponding relevant queries.

2.3.1.4 Miscellaneous

Some other areas try to discover intentions for various reasons; for instance, they are used to deal with the problem of *why end-users accept or reject computers*, to better predict, explain, and increase user acceptance [Davis 1989]. Understanding why people accept or reject computers has proven to be one of the most challenging issues in information systems research [Davis 1989, Swanson 1974]in which mining the intentions plays a key role. These works address the ability to predict humans' computer acceptance from a measure of their intentions, and the ability to explain their intentions in terms of their attitudes, subjective norms, perceived usefulness, perceived ease of use, and related variables.

Kelly et al. propose an approach allowing a robot to detect humans' intentions based on experience acquired through its own sensory-motor capabilities. This takes into account the perspective of the agent whose intent should be recognized [Kelley 2008]. Another work in the domain of home video content analysis [Mei 2005] proposes to capture the user's intention while using camcorders.

2.3.2 Intention Mining Objectives

The objectives of intention mining approaches are classified as follows:

- Discovery: intention mining techniques mainly deal with the intention discovery problem [Strohmaier 2012, Hashemi 2008, Baeza-Yates 2006, Park 2010, Jethava 2011, González-Caro 2011]. Discovery of intentions allows understanding how humans' think, how humans' brains work, identifying the users' intents behind their activities. The aim of intention mining techniques can also be discovering intentional process models from the users' activities or users' logs.

- Recommendation: logs repository and the discovered intentions allow providing recommendations to users at run-time. This is based on the supposed reasoning behind their activities. Several works exist in this category such as [Kumar 2006, Lee 2012].

2.3.3 Intention Mining Techniques

Different techniques have been implemented to discover intentions or to classify them. Some of these techniques are cited as follows:

- Classification techniques: several works use classification techniques to manipulate the Web search queries and to determine intention classes. Some of these techniques are such as, support Vector Machine [Baeza-Yates 2006, González-Caro 2011, Strohmaier 2012], Naive Bayes Classifier [Chen 2002, Strohmaier 2012], FastQ [Jethava 2011] based on Belief Propagation (BP) among others, Lucene similarity measure [Kröll 2009], Complete-link clustering [Sadikov 2010], Cosine similarity [Sadikov 2010], the WEKA [Witten 2005], and the Natural Language Toolkit (NLTK) [Strohmaier 2012]. Some techniques are used to refine the obtained classification, such as Information Gain [Chen 2002], Error-Correcting Output Coding (ECOC) [Baeza-Yates 2006].

- Ontologies techniques: several approaches of intention mining use ontologies techniques. The most common ontologies-based techniques are ConceptNet [Liu 2004] and WordNet [Miller 1995]. ConceptNet is an ontological system for lexical knowledge and common sense knowledge representation and processing. The encoded knowledge in ConceptNet includes lexical concepts and common sense knowledge. The knowledge base is a semantic network consisting of over 1.6 million assertions of common sense knowledge encompassing the spatial, physical, social, temporal, and psychological aspects of everyday life. WordNet is a lexical database for the English language. It groups English words into sets of synonyms, provides short, general definitions, and records the various semantic relations between the sets. The purpose is twofold: to produce a combination of dictionary and thesaurus

that is more intuitively usable, and to support automatic text analysis and artificial intelligence applications. Some research do not use ontologies but predefined categories as the Open Directory Project (ODP) which categorizes websites [Baeza-Yates 2006].

- Text-analysis techniques: Some approaches of intention mining use text-analysis techniques to extract intentions from queries such as, Probabilistic Latent Semantic Analysis (PLSA) [Baeza-Yates 2006], Word unigram and Part-of-Speech Trigrams [Strohmaier 2012], Porter Algorithm [Hashemi 2008], TD/IDF [Park 2010].

2.3.4 Metamodels for Intention Mining

If intentions are referred to as goals, then intentional process modeling refers to modeling the goals underlying the studied processes [Kaabi 2007]. Notations used in intentional process modeling, and therefore intentional process mining are thus goal modeling notations [Nurcan 2005, Yu 2011]. The three salient goal modeling formalisms, i* [Yu 2011], KAOS [Dardenne 1993], and Map [Rolland 1999], that can be used to model processes in terms of users' intentions are described as follows:

- KAOS: the notion of intention in KAOS is considered as *goal*. It is based on a goal diagram where goals are related together through AND/OR decomposition links, which refine high-level goals identified by users' into thinner particle of goals. This refinement requires classifying goals according to their level of abstractions and linking the goals at the same level of abstraction. KAOS proposes to specify the system and its environment by requirements model (instance of a metamodel) to support the goals, actors, and alternatives. KAOS uses goals to specify, to analyze, to negotiate, to document and to modify the systems requirements. This approach supports variability and have a well-structured semantic.

- i*: the i* modeling language aims at analyzing information systems and the environments of organizations to model processes by focusing on the relationships between actors and theirs goals. Actors are autonomous entities with uncontrollable and non-cognizable behaviors. They are different and independent in their ways of reasoning and consequently have diverse goals. i* consists in two main models: the strategic dependency model (SD) and the strategic rational model (SR). The SD describes external relationships between actors, so-called strategic actors. The SR describes the internal relationships between actors. i* is able to assess the functional or non-functional requirements of systems using *soft goals* identified as evaluation criteria; thus it can capture what, how and why a software component is developed.

- Map: the Map formalism introduces explicitly the notion of intention. An intention is a goal that can be achieved by the performance of a process [Rolland 2007]. This model allows describing high-level organizational

and operational intentions. Map metamodel is used to formalize flexible processes. A Map process model (instance of Map metamodel) is presented as a graph, which nodes represent intentions and edges represent strategies. An edge is defined between two nodes where its related strategy can be used to achieve the intended target node. Map metamodel supports variability for the goals and offers the possibility to follow different strategies by focusing on the intentional aspect when enacting methodological processes. Map process models guide the actors by proposing dynamic choices according to their intentions. Concretely, Map metamodel formalizes an intention as a statement expressed in natural language. It usually starts with a verb, which may comprise several parameters, where each parameter plays a different role with respect to the verb [Rolland 2007, Rolland 1999]. Map uses a linguistic approach to define a template for formulating an intention, which is inspired by Fillmore's case grammar [Fillmore 1967] and its extension by Dik [Dik 1989]. The structure of an intention is: Intention: Verb <Target> [<Parameter>]*.

2.3.5 Intention Mining Open Issues

Intention mining field is a new research field; thus the issues that intention mining approaches try to deal with are not as well-known and well-documented as process mining approaches. However, some common issues can be distinguished among the intention mining approaches:

- Ambiguity: the intention is difficult to find since, several intentions in principle match. In other words, it is possible to discover the ambiguous intentions. For instance, an intention discovery technique may confuse a discovered intention with another similar intention [Baeza-Yates 2006].

- Too generic: the intention is vague and many intentions match it. In other words, it is possible to have several intentions that may be categorized into the same class. For instance, a user can perform the similar tasks to fulfill very similar intentions [Baeza-Yates 2006].

  In both cases (ambiguity and too generic) the intentions can be *Several* and they should be categorized into a special category.

- Too specialized: the intention is so specific that will outside the general categories and has to be categorized as *Others*, that is, an unknown category [Baeza-Yates 2006].

2.3.6 Intention Mining Tools

Some approaches use tools to mine intentions. The tool LIBSVM [Chang 2011] is used to implement the Support Vector Machine algorithm [González-Caro 2011]. PennAspect tool [Schein 2001] is used to implement a model to retrieve the users' intentions from the queries and the predefined ODP categories [Baeza-Yates 2006].

Actual workarounds are detected using the Disco tool [Van der Aalst 2011c] that generates models specified using the Business Process Model Notation [Group 2011]. The Natural Language ToolKit (NLTK) is used to manipulate language data and the WEKA data mining toolkit [Witten 2005] for the classification of intentions [Strohmaier 2012].

2.4 Synthesis of Process Mining and Intention Mining Approaches

Tables 2.3, 2.4, and 2.5 indicate panoramas of different process mining and intention mining approaches. These approaches are synthesized in terms of input, users' sources, mathematical model, algorithm, classification techniques, ontologies techniques, objective, and output. Note that the techniques such as, algorithms or ontologies that are not well-known are only specified by a cross.

Table 2.3: An overview of process mining approaches

Research Works	Input	Users' sources	Mathematics Model	Learning	Classification	Ontologies	Objective	Output
[van Dongen 2004a]	Logs	Collective		Extended α-algorithm			Discovery	Petri net
[Dustdar 2005]	Logs	Collective		α-algorithm			Discovery	Petri net
[Van der Aalst 2011b]	Logs	Individual					Recommendation	Petri net
[Maruster 2001]	Traces	Collective		×			Discovery	Petri net
[Van der Aalst 2004b]	Logs	Individual					Discovery	Petri net
[De Medeiros 2006]	Traces	Individual		Genetic algorithms			Discovery	Petri net
[Agrawal 1998]	Logs	Individual		×			Discovery	Acyclic Graph
[Van der Aalst 2004a]	Logs	Collective		α-algorithm			Discovery	Petri net
[Schimm 2004]	Logs	Collective			×		Discovery	
[van Dongen 2005c]	Logs	Individual		×			Discovery	Petri net
[Maruster 2002]	Traces	Collective		×			Discovery	Petri net
[Gaaloul 2005]	Logs	Collective		×			Discovery	Workflow
[de Medeiros 2003]	Logs	Collective		Extended α-algorithm			Discovery	Petri net
[Weijters 2003]	Traces	Collective		Heuristic approach			Discovery	Petri net
[van Dongen 2005b]	Logs	Collective				×	Discovery	Petri net
[van Dongen 2004b]	Logs	Individual		×			Discovery	Petri net/EPCs
[Hammori 2004]	Logs	Collective		×			Discovery	Petri net
[Golani 2003]	Logs	Collective		×			Discovery	Workflow graph
[Van der Aalst 2005a]	Logs	Collective					Conformance	Petri net
[Cook 2004]	Traces	Individual	Finite State Machines	×			Discovery	Petri net
[Herbst 1998]	Traces	Collective	HMMs	×			Discovery	Workflow model

Research Works	Input	Users' sources	Mathematics Model	Learning	Clustering	Ontologies	Objective	Out put
[Hwang 2004]	Logs	Collective		×	×		Discovery	Temporal Graphs
[Mannila 2001]	Traces	Collective	Markov Chain				Discovery	Events
[Herbst 1998]	Traces	Collective	Hidden Markov Model				Discovery	Workflow model
[De Medeiros 2005b]	Logs	Individual		Genetic algorithms			Discovery	Petri net
[van Dongen 2005d]	Logs	Collective		×			Discovery	Petri net/EPCs
[Medeiros 2005]	Traces	Individual		Genetic algorithm	×		Discovery	Petri net
[Weijters 2001]	Logs	Collective		×			Discovery	Petri net
[Greco 2005a]	Traces	Collective		×				Workflow graph
[Schimm 2003]	Traces	Collective		×	×		Discovery	Workflow model
[Van der Aalst 2005c]	Traces	Individual		α-algorithm			Conformance	Petri net
[Herbst 2004b]	Logs	Collective		×			Discovery	Workflow model
[Zhang 2003]	Logs	Collective		×			Discovery	Workflow model
[Van der Aalst 2002b]	Traces	Collective		×			Discovery	Petri net
[Greco 2004]	Traces	Collective		×	×		Discovery	Petri net
[Chen 2003]	Traces	Individual		×			Discovery	Directed graph
[de Medeiros 2005c]	Traces	Individual		α+-algorithm			Discovery	Petri net
[Cook 1998a]	Logs	Individual	Markov Chain	×			Discovery	

Table 2.4: An overview of process mining approaches (following)

Research Works	Input	Users' sources	Mathematics Model	Learning	Clustering	Ontologies	Objective	Out put
[Baeza-Yates 2006]	Logs	Individual			Classification		Discovery	Intentions
[Strohmaier 2012]	Logs	Individual			Classification	×	Discovery	Intentions
[González-Caro 2011]	Logs	Individual		×	Classification	×	Discovery	Intentions
[Kröll 2009]	Logs	Individual			Classification		Discovery	Intentions
[Lee 2012]	Logs	Individual		×			Recommendation	Intentions
[Jethava 2011]	Logs	Individual		×	Classification		Discovery	Intentions
[Ashkan 2009]	Logs	Individual			Classification		Discovery	Intentions
[Shen 2011]	Logs	Individual			Classification	×	Discovery	Intentions
[Baeza-Yates 2005]	Logs	Individual			×	×	Discovery	Intentions
[Yi 2007]	Logs	Individual			Classification		Discovery	Intentions
[Park 2010]	Logs	Individual		×	×		Discovery	Intentions
[Sadikov 2010]	Logs	Individual		×	×		Discovery	Intentions
[Kathuria 2010]	Logs	Individual		×	×		Discovery	Intentions
[Jansen 2007]	Logs	Individual		×	Classification		Discovery	Intentions
[Strohmaier 2009]	Logs	Individual		×	Classification		Discovery	Intentions
[Kumar 2006]	Logs	Individual		×			Recommendation	Intentions
[Outmazgin 2013]	Logs	Collective			Classification		Discovery	Intentions

Table 2.5: An overview of intention mining approaches

Chapter 3

Overview of the Proposed Approach

Contents

3.1 Introduction

So far, in chapter 2 process mining and intention mining have been introduced and detailed with a deep review in the literature. This thesis proposes a novel process mining approach called **Map Miner Method (MMM)**, which aims at inferring underlying users' intentions and strategies from users' traces recorded during the process enactment. Therefore, it is important to clarify the position of MMM regarding the literature of process mining and intention mining fields with respect to some essential aspects: input elements, mathematical model, techniques (algorithms, classification, ontologies), objective, and output elements.

As a first step into this direction, since *intention* play an axial role in the MMM, it is then worthwhile to clarify its nature in the context of this thesis

3.1.1 Intention in Map Miner Method

In MMM scope, an intention is defined as an objective or a motivation to achieve a goal with clear-cut criteria of satisfaction, which users have in mind at a given time that can be fulfilled by the enactment of a process [Soffer 2005]. The intention is of high-level of abstraction in the sense that it expresses what users want to achieve, a state or a result that they want to accomplish. From a technical point of view, the notion of intention in MMM highly depends on the definition of the intention in the Map formalism, since MMM formalizes the intentions by using this formalism. In Map formalism, intentions can be fulfilled with different alternative ways, *i.e.*, strategies, which facilitate the selection of the appropriate alternative for achieving the desired goal [Soffer 2005]. According to this formalism, strategies must lead to intentions. MMM uses this rule to discover the intentions. Indeed, MMM first estimates the strategies, this part will be explained later in this chapter, and it uses the relationships between the strategies and intentions defined in the Map formalism to discover the intentions.

The behaviors of users widely depend on their intentions but not all of these intentions are detectable; thus, it is important to model the level of precision for intentions. MMM generates the high-level intentions to obtain the Map process model and also the low-level intentions to obtain pseudo-Map. These low-level intentions are so-called *sub-intentions*. They are the finest intentional objects, which are associated to a parent intention, and an intention is fulfilled if at least one of its sub-intention is fulfilled.

Indeed, this multi-level topology is due to the deep architecture of the brain. The extensive studies on the visual cortex show each sequence of cortex zones contains a representation of the input and also signals flow from one to the other [Bengio 2009]. In other words, each level of this feature hierarchy represents the input at a different level of abstraction, with more abstract features further up in the hierarchy, defined in terms of the lower-level ones. Therefore, cognitive processes have a deep structure and humans organize their ideas and concepts hierarchically. First, they learn simpler concepts and then compose them to represent more abstract ones [Bengio 2009].

The relationships between users characterize the hierarchy of their intentions. Indeed, the intentions could emanate from an individual user or a group of users when delegating a task from a user to another one or to a group of users (nested delegation of the tasks). In this case, related intentions are also delegated. However, these relationships are not detectable directly by event logs. MMM assumes that each user tends to fulfill own intention in a given process.

3.1.2 Map Miner Method Process Model Formalism

Process mining approaches mainly use Petri nets or BPMN to design process models. These formalisms have a rigid structure and they do not support unstructured processes when a sequence of activities vary in different situation or when a process

is executed in non-sequential ways. Moreover, they cannot support variability and flexibility, which makes impossible the adaptability to the context of each user.

The literature suggests several intentional model formalisms, such as KAOS [Dardenne 1993], Map [Rolland 1999], i* [Yu 2011, Dardenne 1993], Tropos [Bresciani 2004] and GRL [Amyot 2009] among others.

KAOS has a rigid task-decomposition - Refinement cannot be separated from OR decomposition and AND decomposition, which introduces artificial complexity in the goal hierarchy; therefore, modeling complex intentional processes is difficult. Another issue of KAOS is to sort goals according to their level of abstraction and relate goals when they belong to the same level of abstraction. Furthermore, KAOS is less involved in the intentional aspect of information systems actors since, it does not support strategic/organizational goals.

i* modeling language has an operational semantic for the tasks but not for the goals and it is not used to model strategic goals. i* is not designed to be a variable framework therefore, it does not afford a high flexibility. The main difficulties with i* is that it lacks (i) systematic goal refinement mechanisms (all the goals are defined in the same level), and has no (ii) goal-strategy couple to help clarifying the multiple ways in which a goal can be achieved. i* models find their limits when the situation gets complex.

Note that Tropos is an agent-oriented software engineering methodology that includes i*; GRL is an intentional modeling language based on a subset of i*. Thus, they both have the same limits as i*.

In short, the aforementioned formalisms do not use an intention as an integral part of the model due to the fact that these models internalize how the process is performed and externalize what the process is intended to accomplish in the goal [Dietz 2005, Rolland 2007].

The Map modeling language is an intentional process metamodel used to formalize flexible processes. The intentional Map metamodel has been introduced in the information systems engineering domain [Rolland 1999] and was validated in several works: requirement engineering [Prakash 2006], method engineering [Kornyshova 2007], and enterprise knowledge development [Barrios 2004]. Further, it has proved to be effective to specify business processes, user requirements, systems functionality, engineering methods, software engineering processes, etc [Rolland 1999].

Next section will explain the Map metamodel and the Map process model.

3.1.3 Map Metamodel

A Map metamodel allows representing strategy-oriented processes. The Map metamodel permits specifying processes according to users' intentions, and different ways to achieve them, so-called *Strategies*. The strategies in the Map metamodel provide the means to capture variability in intention achievement. Thereby confronted to a specific situation and a particular intention of the user, the process model reveals the alternative strategies to follow the intentions, and the intentions to pursue.

Figure 3.1 illustrates the Map metamodel.

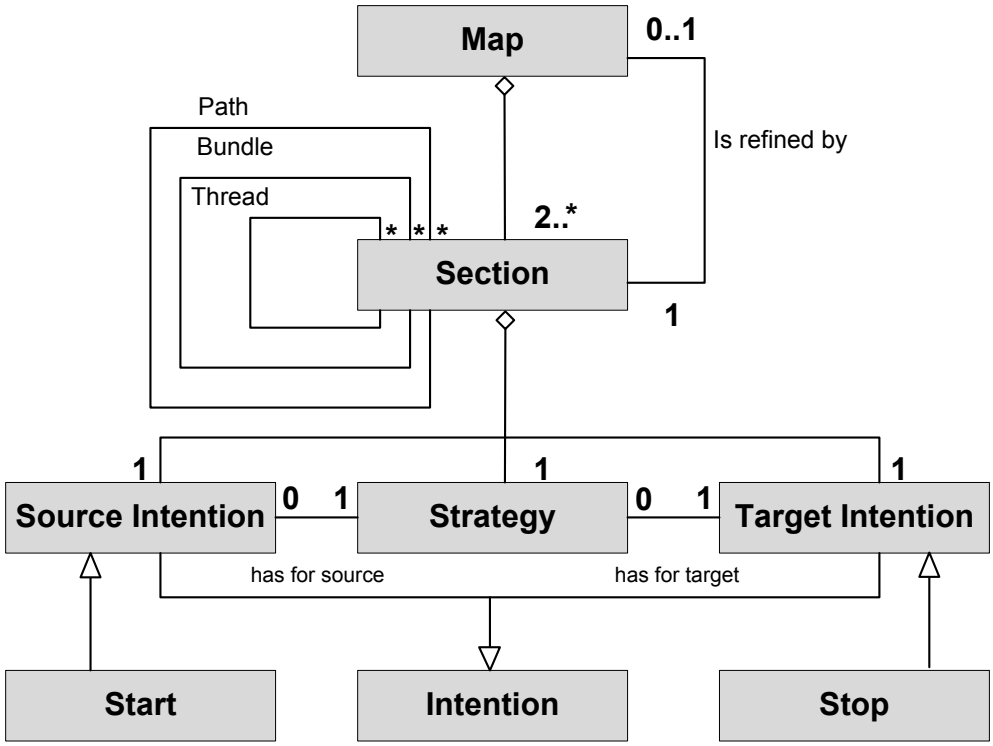

Figure 3.1: The Map metamodel [Rolland 2007]

3.1.3.1 Map Process Model

Map process model is an instance of the Map metamodel presented as a directed graph (see Figure 3.2). The nodes specify *intentions* and the edges specify *strategies*. Map process model provides a navigational structure that supports the dynamic selection of the next intention to be achieved and the appropriate strategy to achieve it. This Map process model is used in Section 4.5 for method validation.

According to this model, users can select ten strategies to fulfill four intentions. These intentions are **Start**, **Specify an entity**, **Specify an association** and **Stop**. Each edge represents a strategy that a user can select to fulfill an intention (represented as a node) according to his/her situation. For instance, if the current situation is **Start** and the user's intention is to **Specify an entity**, there is only one strategy *by completeness of the model* to fulfill this intention. When the current situation and the intention is **Specify an entity**, there are four strategies *by completeness, by generalization, by specialization, by normalization* to fulfill the

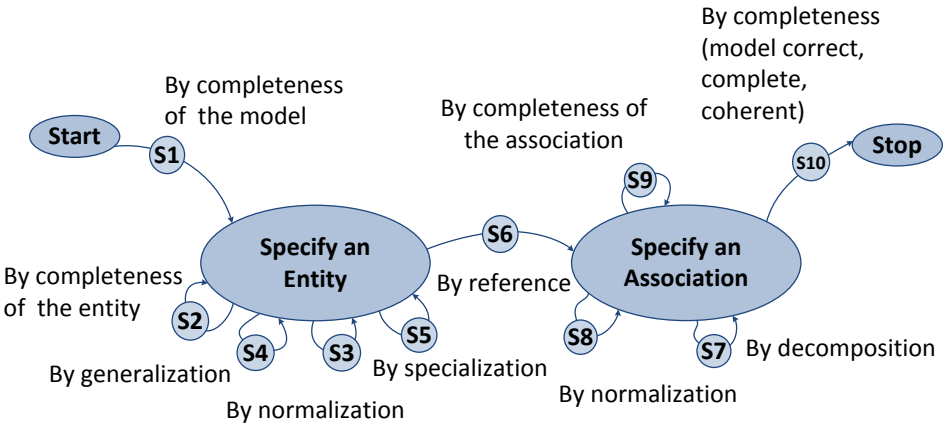

Figure 3.2: Prescribed Map process model for construction of E/R diagrams [Assar 2000]

same intention. It is possible to continue progressing in the process by selecting the strategies that lead to the considered intentions but once the **Stop** intention is achieved, the enactment of the process is finished.

3.1.3.2 Map Section

A *section* is a key element of a Map process model, which can be seen as an assembly of sections [Rolland 2007]. A Map section highlights a consistent characteristic of the system that stakeholders want to implement through some functionality. A Map process model is composed of at least two *sections*. A section can be refined by another Map process model. It allows decomposing a section to detail it. A section is composed of <**Source Intention**, **Target Intention**, *Strategy*> for linking the source to the intended target by a strategy. There are three types of links between sections [Rolland 2007]: thread, path, and bundle (see Figure 3.1). Hereafter, they are briefly described:

- Multi-thread: two sections are multi-thread if they have the same source and target intentions and different complementary strategies such as, strategies S_2 and S_4 in Figure 3.2.

- Multi-path: represents a sequence of several sections having the same source and target intention but defining different paths in the Map. A possible path can be the sections: < Start, Specify an entity, By completeness of the model>, <Specify an entity, Specify an association, By reference>, <Specify an association, Stop, By completeness>.

- Bundle: two sections form a bundle if they have the same source and target intentions but different exclusive strategies such as, strategies S_4 and S_5.

3.1.4 Map Process Model Advantages

In MMM framework, Map process model is chosen rather than other intentional process models for several reasons:

- The Map topology: combining intentions and strategies, at different abstraction levels, allows handling large-scale and complex processes [Rolland 2005b]. Indeed, during its enactment, a process is not limited to linear activities; users, according to their context, have a variety of choices to perform an activity. Map process models (instances of Map metamodel) guide the users by proposing dynamic choices according to their intentions. The Map strategies can be executed non-sequentially and be followed until the fulfillment of intentions. Thereby, Map process models offer a better adaptability to the context of each user. Moreover, fulfilling a specific intention with a particular strategy can be related to a specific guideline defining the activities to perform.

- The Map formalism: it is intuitive and easy to apply and to understand. This formalism is particularly suitable for representing unstructured processes, whose sequence of activities may vary according to the situations, or processes including variability whose sequence of activities is selected at run time depending on the situation at hand.

- The Map metamodel supports process variability and flexibility by defining different strategies to fulfill a given intention. New information systems need to meet the multi-purposes of several organizations and not only a single one [Rolland 2007]. Therefore, capturing variability among the intention is essential. Integrating strategies as an integral part of a process model changes a system from a mono-purpose to a multi-purpose. This property is unique to the Map process model.

- Last but not least, the Map process model drives the process by pursuing the users' intention to achieve a goal. As a result, goals are explicitly represented in the process model along with the alternative ways for achieving them. Thereby, it allows variability in goal achievement and facilitate the selection of the appropriate alternative for achieving the goal during process enactment.

3.2 Contributions of this Thesis

MMM derives the essential principle of process mining field, which means mining event logs to find users' behaviors pattern within a process. However, contrary to process mining approaches, MMM models users' behaviors in terms of their intentions and strategies. It adopts intention-oriented process model to design the

processes while process mining approaches specify users' behaviors in terms of sequences of tasks and branching. Indeed, intention-oriented process models are an adequate formalism to guide users through the enactment of their activities since, the fundamental nature of processes is mostly intentional [Rolland 1998a]. They allow modeling processes in a flexible way (the notion of sequence does not exist), variability can be introduced (alternatives path can be followed, different strategies can be used to achieve the same intention). This kind of process model allows more creativity than process mining approaches [Rolland 2005b].

Process mining approaches adopt an operational view focusing on how the process is performed. In contrast, MMM focus on what the process is intended to achieve, thus providing the rationale underlying the process, *i.e.* why the process is performed. In other words, MMM follows the humans' intentions during the process enactment as the force that drives the process, and concentrate on what the process must do, *i.e.*, on its rationale. Intentions are fundamental to human behavior, playing a central role in both its enactment and its understanding [Chulef 2001]. The intentions of an individual as well as his/her interactions with computer systems are essential for understanding and predicting the behavior in which individuals engage.

Due to the intention-oriented process models properties, some problems identified in process mining such as hidden tasks, duplicate tasks and loops do not hinder the intentional process models discovered by MMM. Indeed, the activities are of less importance with a representation on a higher level (*i.e.*, intentions and strategies). In contrast, in process mining approaches activities play an axial role in the process modeling, which makes the discovered process models vulnerable in the presence of a hidden or duplicated activity in the data logs. MMM framework supports duplicated activities in the sense that a given activity may exist in different strategies/intentions. Regarding hidden activities, they do not impact strongly the discovered process models, since even though an activity is missing, MMM can estimate the strategies and discover the intentions. This is due to the nature of strategies and intentions comprising several activities; if one or even several activities are missing, others can indicate the right strategies/intentions. Whereas loops are often a difficult problem to handle in process mining, they are a usual concept in MMM as in Map process models, a section can be enacted several times, until the desired intention is achieved.

3.2.1 Map Miner Method Input Elements

The input of MMM, like several process mining approaches, is the temporal set of users' activities - interactions of users with a information systems tool during a time slice $\Delta = t_N - t_0$, where t_0 is the beginning of the activity performance and t_N is the end. During a time Δ one or several sequences of activities are recorded by a tool. These sequences are a *trace* of activities for that user. The input of MMM, like several process mining approaches, is the temporal set of users' activities - interactions of users with a information systems tool during a time slice

$\Delta = t_N - t_0$, where t_0 is the beginning of the activity performance and t_N is the end. During a time Δ one or several sequences of activities are recorded by a tool. These sequences are a *trace* of activities for that user.

In the MMM, the entire users' traces that occurred during a process enactment are analyzed. Note that a process in information systems context is defined as a sequence of activities linked to each other by a common goal [Rozinat 2010]. Therefore, there are strong correlation and dependency between users' traces and they cannot be considered as a single, independent and uncorrelated entity. A sequence of activities contains more enriched information about users' intention than a single activity, and this from both semantic and abstraction level points of view. Indeed, analyzing a sequence of activities allows determining the high-level intentions (*e.g.*, organizational goal), while analyzing single activities leads to less informative low-level intentions, also called basic intentions or action intentions, which are closer to activities than intentions.

The inputs of intention mining approaches are the single query or activity (individual entities) and they do not take into account the logical dependencies between each input element (see Section 4.2.1 for further information).

Process mining and intention mining approaches are applied for the logs/traces of one or many users. The MMM analyzes the traces of a group of users realized while enacting a process.

3.2.2 Map Miner Method Mathematical Model

The mathematical model used in MMM is Hidden Markov Models (HMMs) [Rabiner 1989]. Among the techniques to model different aspects of humans' behavior [Gray 1992], Hidden Markov Models (HMMs) have been proven to be appropriate for modeling the real world process, particularly unobservable cognitive processes [Hayashi 2003, Hoey 2007]. HMMs are stochastic finite automaton and a special kind of Bayesian Network [Friedman 1997]. HMMs offer all the properties of the stochastic model in both statistical and probabilistic framework. HMMs have been used in a wide variety of contexts and have proven valuable in diverse fields such as speech recognition [Juang 1991, Gales 1998, Rabiner 1989], financial data prediction [Zhang 2004], signal processing [Kil 1996], generic temporal data clustering [Li 1999] and later applied widely in the bioinformatics field [Martelli 2002, Delorenzi 2002]. Several reasons motivate this choice, which are described hereafter.

The real-world processes generate a sequence of signals - observable discrete or continuous symbols (representation of a physical phenomenon) [Rabiner 1989]. Existing physical signals in nature are generally analog or continuous. A signal is continuous when it is a continuous function of time, such as a speech, an image, etc. Digital or discrete signals are set of values collected at regularly spaced instants, such as alphabet symbols. The observed signals can be modeled to explain and characterize their occurrence. This modeling can be used later to identify or recognize other sequences of observations.

The modeled observations could have different forms: linear or non-linear, time-variant or time-invariant. A linear system has only one dimension (length) and models the observed symbols as output. A non-linear system has more than one dimension and the output is not directly proportional to the input. The time-invariant system output does not explicitly depend on time, contrary to a time-variant system. By analogy, since traces are random variables, the realization of traces of activities is not deterministic and follows a stochastic process. Moreover, the multi-levels topology of users' activities, strategies and intentions are a time-variant and non-linear system. Due to the nature of input data - the users' activities - and output data - the users' intentions and strategies - a stochastic model should be chosen because it allows:

- Describing a given system as a theoretical statistical model, *e.g.*, knowing the significance of the observed sequences,
- Analyzing the observed sequences over time,
- Modeling and predicting the latent states of these observed sequences,
- Extracting the characteristics of observed and latent sequences.

Intentions express what users intend to perform during the enactment of a process [Rolland 1999]. More precisely, the enactment of a process is a sequence of intentions which are fulfilled by several **strategies**. The **activities** executed by a user to fulfill his/her intention are a consequence of the target intention and the selection of the strategy to fulfill it. These intentions, strategies, and activities represent the top-down reasoning and acting structure of cognitive processes of humans' brain. However, only the low-level part of this structure, *i.e.*, users' activities, is observable. The middle and high-level part, respectively strategies and intentions, are abstract notions and therefore unobservable directly. According to the fuzzy mechanism of a cognitive process, an intention causes one or more activities to be performed at time t. A set of activities that is realized to fulfill a given intention is a strategy. Furthermore, the topology of HMMs is appropriate to model hidden users' strategies from observable users' interactions with the system. On the one hand, this is similar to the HMM structure, *i.e.*, hidden states and observed process. On the another hand, according to the HMM assumption an event can cause another event in the future and not in the past. This assumption fits perfectly the prospective structure of intentions since, an intention can cause another intention in the future but the opposite is not true.

Some process mining approaches use HMMs to reflect the process model on another level [Herbst 1998]. Their works combine HMMs with another algorithm to extract workflow models. It does not address the intentional dimension of processes. Van der Aalst considers HMMs as versatile and relevant for process mining but HMMs unsupervised approaches are complicated, since: (a) there are computational challenges due to time consuming iterative procedures, (b) the number of states (as algorithm inputs) should be known, (c) the result of HMMs is not

very understandable for the end-user [Van der Aalst 2011c]. Although the first point is a valid statement, if one overcomes the other issues, it seems worthwhile to use unsupervised learning of HMMs that is proven to converge to a local optimum [Rabiner 1989]. Regarding the second point, there are different techniques to find the number of states (see Chapter 4). Finally, unlike many other works, in this thesis HMMs are used to estimate users' strategies. The notion of strategy has an understandable semantic for the end-user. Another work uses HMMs as a conformance checking technique by measuring similarities between Markov models using a distance metric [Rozinat 2008b]. It enables the stakeholders to evaluate the quality of mined processes. The workflows modeled in Petri nets are mapped to HMMs but the hidden states of processes are not taken into account. This thesis proposes to model the hidden states of an HMM as users' strategies. However, none of the aforementioned techniques considers the hidden states of HMMs as humans' cognitive process (*e.g.*, users' strategies). The proposition of this thesis to thereby model an HMM makes a huge difference of the application of HMMs on the logs to model users' behaviors.

3.2.3 Map Miner Method Techniques

The process mining approaches use many well-known or particular algorithms, classification or clustering techniques, ontologies-based techniques, etc. For instance, they mainly use α-algorithm to model event logs. However, α-algorithm cannot handle noise and certain complicated routing constructs of workflow nets such as loops and long-term dependencies, particularly during complex situations [Rozinat 2010]. The intention mining approaches use mainly classification techniques to classify a single input into a class of intention. The choice of classification techniques seems accurate since, these works try to infer an intention related to a query and then classify it into a category. They do not consider users' intentions as a part of the process enactment. Whereas, in this thesis, due to the multi-levels structure of MMM (users' activities, strategies and intentions), there has to be a mathematical model to set up, at least, a two-level topology for the users' activities and strategies.

Two tailored algorithms are developed to reconstruct the Map process models in different levels of precision. The developed algorithms are the **Deep Miner** algorithm and the **Map Miner** algorithm. They are briefly explained in Section 3.2.4.

3.2.4 Map Miner Method Objectives

The aim of process mining approaches is to discover the process models, to check the conformance between the prescribed process model and the discovered one, and to improve the process models. Most approaches of process mining deal with process discovery challenge [Rozinat 2010]. However, all of these objectives are based on discovering the users' tasks sequences (activities).

MMM tackles mainly the process discovery challenge, which is essential for conformance checking and process enhancement. Therefore, MMM takes as inputs

users' traces to generate an intentional model based on users' intentions and strategies, which differentiates it from process mining approaches in terms of outputs.

The MMM purposes are then defined below:

- Map Discovery: MMM aims at inferring underlying users' intentions and strategies with respect to the Map formalism through users' activities recorded during process enactment. This allows constructing Map process model. Chapter 4 discusses in detail the different stages of MMM. Figure 3.3 depicts an overview of MMM. The phases of MMM are described as follows:

  - Estimation of parameters: this phase aims at estimating the HMMs parameters. The parameters of an HMM must be estimated by supervised or unsupervised learning:
    * Supervised learning allows estimating the parameters of an HMM by Maximum Likelihood Expectation. It takes as inputs users' activities along with related strategies and the outputs are transition and emission matrices. Therefore, the supervised learning requirement to the traces of strategies makes it a costly technique, since the users' should label their activities. However, supervised learning has some advantages: it allows validating the proposed hypothesis from a practical point of view. The supervised learning also allows checking the conformity of the prescribed models and the discovered models.
    * Unsupervised learning allows estimating the parameters of an HMM by Baum-Welch algorithm. It takes as inputs users' activities and number of strategies and the outputs are transition and emission matrices. Unsupervised learning has some advantages: it do not require the traces of strategies related to the traces of activities. This makes it an easy technique to use and to automate the method. In addition, unsupervised learning is more efficient than supervised learning. In other words, the results obtained by unsupervised learning are more likely to reflect actual relations between traces than supervised learning.
      HMMs use transition and emission matrices to model observed activities in terms of hidden strategies.
  - Deep Miner algorithm: this algorithm is developed to discover sub-intentions and thereby reconstructs pseudo-Maps. Deep Miner algorithm uses the new metric of fitness and precision, which has the interesting property of taking into account both fitness and precision to optimize the Map process models. Deep Miner uses Map formalism to define the topology of a Map.
  - Map Miner algorithm: this algorithm groups sub-intentions into high-level intentions and thereby reconstructs Map process model.

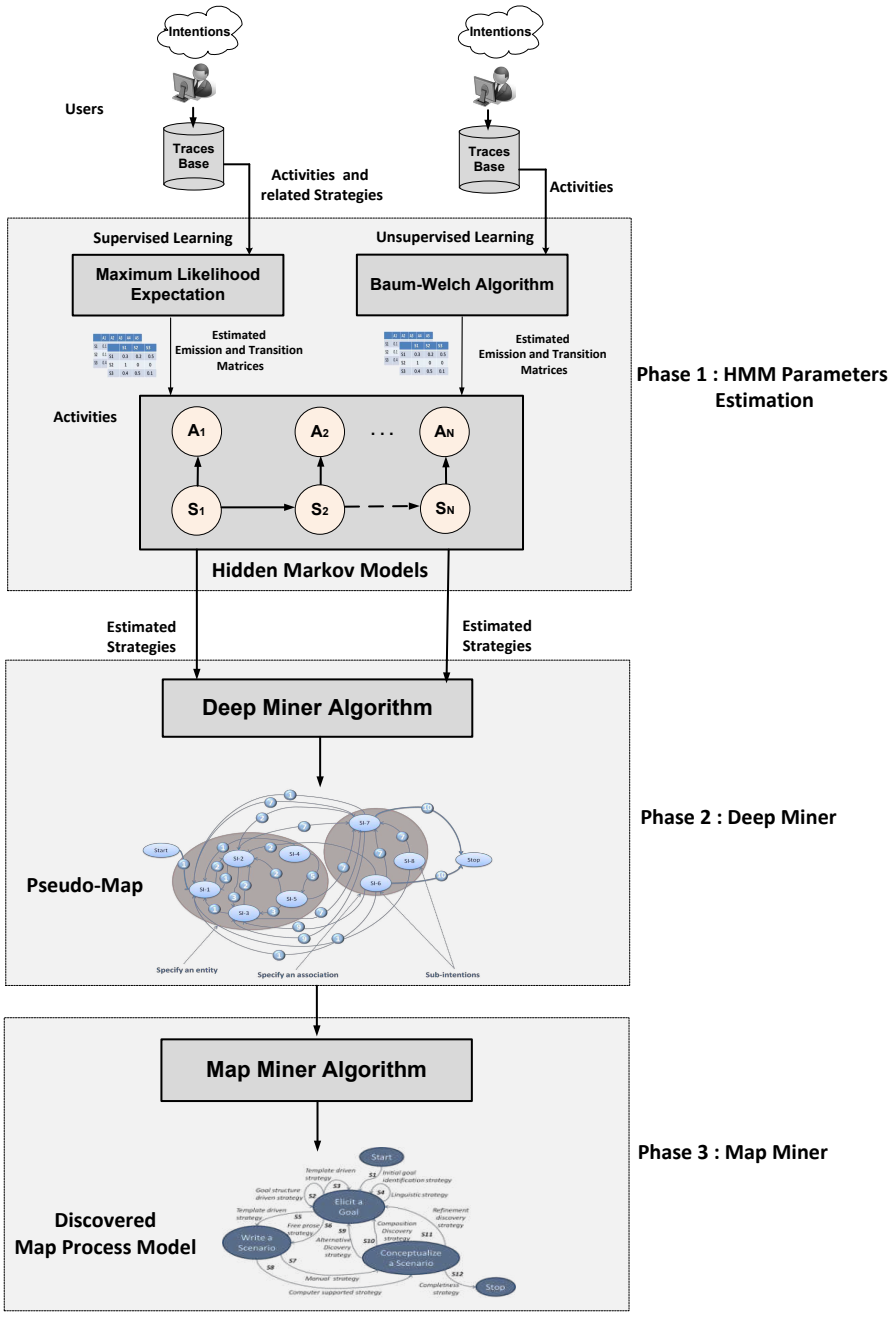

Figure 3.3: Overview of Map Miner Method Framework

To do so, Map Miner algorithm uses a clustering algorithm, *K-means* [Hartigan 1979] to group the sub-intentions into the intentions. As explained earlier, this multi-level topology is due to the deep architecture of the brain and the fact that humans organize their ideas and concepts hierarchically. An algorithm is also developed to rebuild the

Map process model from clustered sub-intentions.

- Discovery of Map path: the event logs repository from the enactment of process allows finding the most likely sequence of strategies related to a sequence of activities. This sequence of strategies is a path in the Map process model (see Figure 3.4).

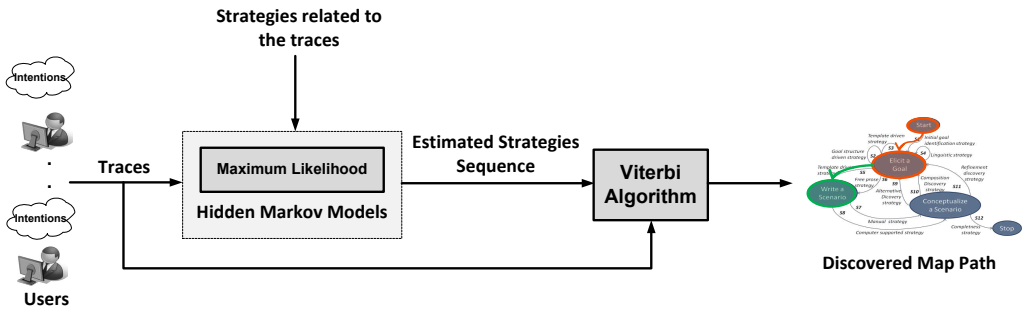

Figure 3.4: Discovery of Map path with supervised learning

3.2.5 Map Miner Method Output Elements

The outputs of process mining approaches are generally process models or workflows. The output of intention mining approaches are single intentions. Whereas in MMM context, the output is a Map process model discovered from traces.

3.2.6 Map Miner Tool

Map Miner is a tool developed to automate the construction of Map process models. It is a portable tool since it is implemented in Java; thus, it can be plugged into other platforms such as ProM. The Map Miner tool can be customized to generate the tailored Map process models. This is possible by adjusting different inputs parameters. Map Miner has a simple interface, which enables users readily manipulating the parameters, the input files and allows visualizing the Map process models in different levels of abstraction. Map Miner is also equipped with a DataBase Management System (DBMS). This embedded database allows importing directly the traces from another database. Map Miner tool is described in chapter 6.

3.3 Summary of the Position of the Proposed Approach

Table 3.1 gives a summarized synthesis of process mining, intention mining approaches and MMM over different aspects. As found in this table, the input of process mining and intention mining approaches are either users' traces or users' logs. The input of MMM is the traces of users, which contain users' activities. While

the process mining approaches analyze data from either individual user or the group of users. The intention mining approaches make an individual processing. MMM analyzes data from group of users enacting a given process. Some process mining approaches use mathematical models to design the logs. The intention mining approaches generally do not use mathematical model. MMM uses also a mathematical model, which diverges from other approaches in the proposed conception. The process mining approaches use mainly α-algorithm and also many other particular algorithms developed for each approach. The intention mining approaches also use algorithms designed for special needs of a particular approach or the algorithms related to machine learning techniques. The process mining and intention mining approaches use the classification (or clustering) techniques to classify the extracted logs. MMM also uses a clustering technique to classify sub-intentions into high-level intentions. Whereas the process mining approaches rarely use ontologies (see Chapter 2), the approaches of intention mining use them more often. MMM does not use ontologies to infer the names of strategies and intentions, this part is manually done. The objectives of process mining is discovery, conformance checking, enhancement, and recommendation. So far, the intention mining approaches mainly have focused on intention discovery and providing recommendations for the end-user. MMM mainly focuses on Map process model discovery. The outputs of the process mining approaches are activity-oriented process models or workflow models. The intention mining approaches outputs are mainly single users' intentions. They do not generate intentional models. MMM generates intention-oriented process models (Map process model).

Research Works	**Input**	**Users' Sources**	**Mathematics Model**	**Learning**	**Classification**	**Ontologies**	**Objective**	**Output**
Map Miner Method	Traces	Collective	HMMs	Baum-Welch, MLE, Viterbi	K-means		Discovery	Map process model
Process Mining approaches	Logs, Traces	Collective, Individual	HMMs	α-algorithm, Many others	×	×	Discovery, Recommendation, Conformance, Enhancement	Petri nets, BPMN, EPCs, etc
Intention Mining approaches	Logs, Traces	Individual		×	×	×	Discovery, Recommendation	Intentions

Table 3.1: Summary of process mining, intention mining approaches, and Map Miner Method

Chapter 4

Proposed Method: Map Miner Method

Contents

This chapter presents in detail the different parts of the proposed method Map Miner Method (MMM). The method is exemplified through a single example that is presented in the next section.

4.1 Presentation of the Example

A single example will be used throughout this chapter to exemplify the validation of the proposed method step-by-step. To do so, a tailored experiment was conducted

with 66 Master students in computer science of the University Paris 1 Panthéon-Sorbonne. Table 4.1 presents the profile of the students. In this example, the students were asked to follow a prescribed Map process model (prescribed Map in short) presented in the Figure 3.2. This prescribed Map guides users through the creation of Entity/Relationship (E/R) diagrams (The original Map was proposed in [Assar 2000]). A web application developed in HTML, PHP, JavaScript and MySQL to record the users' traces. This application records traces of executions carried out by the students during the creation of their diagrams in a database.

Total	**Average age**	**Sex**		**Master degree**	
		male	female	1st year	2nd year
66	24,4	49	17	48	18

Table 4.1: Profile of the Students

4.2 The Products of the Method

This section gives the definition of the products used and produced by the method.

4.2.1 Input of MMM

Raw data is hidden in all kinds of data sources. Nevertheless, this does not mean that data is well-structured. The event logs are typically distributed over data sources and often some efforts are required to filter the relevant data. In other words, the most interested parts of event logs for mining process (*i.e.*, users' activities) should be extracted from non-filtered event logs. Indeed, non-filtered event logs contain not only the users' activities but also the other information depending upon the context. The data that can be used as input of MMM must be a well-structured and filtered (*i.e.*, users' activities). The process of filtering the relevant data involves extracting users' activities from non-filtered event logs, which contains users' identifier, users' activities, and timestamps, etc. This stage could be done manually or by using code Snippets. In this thesis this part is done manually. Table 4.2 depicts a fragment of non-filtered event logs, for the example.

UserID	**TraceID**	**Timestamps**	**MapSID**	Activities	...
45	7	31/10/2012 14:54:00	1	Create entity	...
22	1	31/10/2012 15:14:00	4	Create generalization link	...
61	2	23/10/2012 08:54:00	10	Create attribute	...
12	8	31/10/2012 14:54:00	7	Create association	...
45	7	23/10/2012 09:41:00	2	Link attribute to entity	...
38	4	23/10/2012 09:45:00	8	Delete association	...
...	...	...	...	...	...

Table 4.2: A fragment of the trace for the example

This table illustrates the typical information related to a single process in logs data. Each line represents an event containing data such as, UsersID, TraceID, Timestamps, MapSID (MapSectionID), Activity. These event logs can be enriched with some more information, *i.e.*, with annotations of the user while performing an activity in the system or after the completion of the activity, which give information about the rationale underlying the activities. For instance, in the example, the students annotated their executions. Some additional information is also recorded per event. For example, MapSID is recorded for each event log in this particular example (see Section 3.1.3.2 for the definition of the Map section). The event also contains a userID that identifies the user who is performing the activity. Moreover, all events have timestamps, which means date and time information such as, '23/10/2012 09:41:00'. This information plays a key role for MMM since it determines when exactly a given user performs an activity and what are his/her next activities. This allows knowing about the ordering and dependencies of activities, which are essential for the modeling phase. These temporal set of activities performed by a given user are the *trace* of activities of that user. A trace of activities is result of interactions of a user with an information system tool during a time slice $\Delta = t_N - t_0$, where t_0 is the beginning of the activity performance and t_N is the end. During a time Δ one or several sequences of activities are recorded by a tool. The properties of trace in MMM context are given as follows:

- A trace consists of events such that each event relates to the enactment of an activity.
- Events within a trace are ordered.
- Traces may belong to different instances of a process.
- The events within a user's trace must be ordered in terms of timestamps to discover causal dependencies in the process enacted by users.

The input of MMM is the users' traces.

4.2.2 Users' Activity

The activities are the interactions of a user with an information system software recorded by a tool. To fulfill a given intention, users have to carry out one or several activities through an information system tool. There are different kind of activities with respect to the users' relationships, intentions and environments: intentional or accidental activities. The intentional activities result from an intentional decision made by a user while accidental activities are outcomes of a non-intentional user's behavior. The accidental activities are probably produced by mishandling of keyboard, internal system tools interactions, errors of OS, errors of IDE, etc.

In MMM, the assumption is that an activity is the result of interactions of a user with an information system. Thus, there is neither task delegation between users nor nested-intention. Therefore, all kind of recorded activities are performed by

a user on information system with his/her own intentions. This definition encompasses intentional actions of users. The non-intentional and accidental activities are overlooked by MMM. Table 4.3 represents a fragment of activities related to the example. As it can be noted in this table, the activities appear with an index such as a_1, which represents the 'Create an entity' activity.

Index	**Related activities**
a_1	Create an entity
a_2	Link attribute to entity
a_3	Create an association

Table 4.3: A fragment of activities for the example

In MMM context, the following associations and cardinalities exist between activities and traces within every process:

- Every process may have an arbitrary number of activities.
- Every trace belongs to precisely one process.
- Every event refers to precisely one trace.
- Every activity has attributes such as a name that makes it human-understandable.

4.2.3 Strategies, Intentions

The definition of a *strategy*, a *sub-intention*, and an *intention* in this thesis, are given in Chapter 3. Here, they are investigated from a technical point of view.

A strategy is a set of activities performed together. These activities are intercorrelated and dependent to each other. For instance, in Table 4.2.3 the set of activities a_1 and a_3 constitutes the strategy *by generalization* and the set of activities a_1 and a_4 constitutes the strategy *by specialization*. As the activity a_1 is defined common to two strategies, which means an activity can be mixed with other activities to constitute different strategies. Note that the order of strategies is not important in MMM. For instance, to select the strategy *by specialization* the user can perform first the a_1 activity and then the a_4 activity or vice versa.

	Strategies	**Related Activities**	**Activities Index**
4	by generalization	Create entity, Create generalization link	a_1, a_3
5	by specialization	Create entity, Create specialization link	a_1, a_4

Table 4.4: Strategies and related activities

According to the Map semantic, the strategies must lead to the intentions. Therefore, once the topology of the strategies is discovered, intentions can be inferred as the targets of the strategies. For instance, the strategies *by generalization* and *by specialization* both lead to the intention **Specify an entity** (see Figure 3.2). Consequently, if these strategies are discovered, it is possible to find the node to which they lead, which means the intention **Specify an entity**. The intentions are shown by the large nodes in Figure 4.1.

4.2.4 Pseudo-Maps and Sub-intentions

By using MMM with the traces of the example it occurs that the intentions that MMM discovers have a higher degree of precision in the sense that they are more precise than the prescribed intentions. The notion of sub-intentions does not exist in the Map semantic. This led us to creation of a new concept in Map formalism: **sub-intentions**. Therefore, this notion can be considered as an extension of the Map metamodel. As defined in Chapter 3, sub-intentions are the finest intentional objects, which are associated to a parent intention, and an intention is fulfilled if at least one of its children sub-intention is fulfilled. They are shown with small nodes in Figure 4.1 and indexed by $SI_1, SI_2, \cdots$. A process model containing the sub-intentions, the strategies, **Start** intention and **Stop** is called a **pseudo-Map**. MMM first discovers the sub-intentions and then it clusters sub-intentions into high-level intentions to obtain a Map process model.

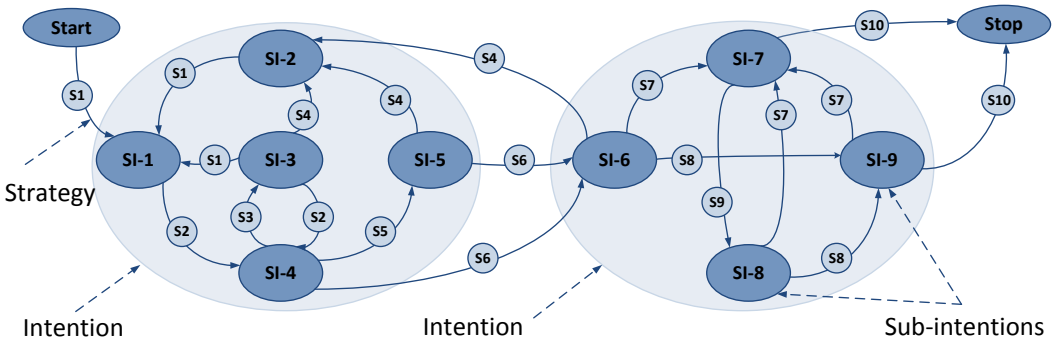

Figure 4.1: Strategies, sub-intentions, and intentions

The relationships between users' activities, strategies, and intentions are defined as: performing one or several activities relates to the enactment of a strategy and consequently to the fulfillment of an intention. The relationships between users' activities and strategies are defined in the matrices. This is explained in the next section.

4.2.5 Transition and Emission Matrices

The relationships between users' activities and strategies as well as the correlation within strategies are the parameters of a Hidden Markov Model (HMM), which can be estimated from data and expressed by two matrices:

- Emission matrix: it describes the occurrence of activities in strategies of a process. In other words, emission matrix elements allow knowing how many times a given activity appears in a given strategy. The related probabilities are called emission probabilities.
- Transition matrix: it describes the transitions probabilities between all the strategies in a process. The transitions are the changes of strategies from one to another and the related probabilities are called transitions probabilities.

The probability distribution of the initial state (strategy), defined by π, must also be initialized. It is essential to compute the emission and transition matrices. For instance, let us assume the initial probability of the transition matrix is: $\pi = \{1, 0, 0, 0\}$. Each value corresponds to an intention of the example. The first value expresses the probability that a user at time step t has the intention **Start** is one. The second value expresses the probability that a user at time step t has the intention **Specify an entity** is zero, and so forth. This assumption is rational since according to the prescribed Map (Figure 3.2) the intention **Stop** cannot be achieved unless **Specify an association** is already achieved and this latter can only be achieved if **Specify an entity** have been achieved earlier.

To illustrate the relationships between activities and strategies, let us consider a case with three strategies $\{S_1, S_2, S_3\}$ and three activities $\{a_1, a_2, a_3\}$. The MMM parameters are described by the probability distribution of the initial strategy and by two matrices of emission and transition:

A 3×3 matrix for the emission probabilities of activities for each strategy, denoted by $\mathbf{E}$:

$$\mathbf{E} = \begin{pmatrix} \mathbf{E}_1(a_1) & \mathbf{E}_1(a_2) & \mathbf{E}_1(a_3) \\ \mathbf{E}_2(a_1) & \mathbf{E}_2(a_2) & \mathbf{E}_2(a_3) \\ \mathbf{E}_3(a_1) & \mathbf{E}_3(a_2) & \mathbf{E}_3(a_3) \end{pmatrix}$$

A 3×3 matrix for the transition probabilities of strategies denoted by $\mathbf{T}$:

$$\mathbf{T} = \begin{pmatrix} \mathbf{T}(S_1, S_1) & \mathbf{T}(S_1, S_2) & \mathbf{T}(S_1, S_3) \\ \mathbf{T}(S_2, S_1) & \mathbf{T}(S_2, S_2) & \mathbf{T}(S_2, S_3) \\ \mathbf{T}(S_3, S_1) & \mathbf{T}(S_3, S_2) & \mathbf{T}(S_3, S_3) \end{pmatrix}$$

For instance, the value associated to $\mathbf{T}(S_1, S_2)$ expresses the probability of transition from the strategy S_1 to the strategy S_2 or the value associated to $\mathbf{E}_3(a_2)$

expresses the probability that the activity a_2 appears in the strategy S_2 (the second column represents strategy S_2). Note that the auto-transitions (loops) such as $\mathbf{T}(S_1, S_1)$ signifies the probability that a user continues to perform the activities related to the strategy S_1.

Note that the emission and transition matrices must be initialized for the first execution. This initialization is arbitrary. In this thesis the initialization of these matrices are realized as follows:

$$\mathbf{E} = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$

$$\mathbf{T} = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

In the emission matrix, each element of the matrix takes a value equal to $\frac{1}{Numberofcolumns}$. The rationale behind this initialization is that it is assumed that all the activities existing in the traces can be presented in each strategy.

For the transition matrix, the procedure of initialization is different. The diagonal elements have higher values than other elements. The diagonal values represent the transitions of each strategy to it-self. In other words, the time taken to realize completely the strategy. This is due to the structure of the strategies that are composed of activities. Indeed, the realization of a given strategy involves the realization of all the activities that belong to it. The realization of a strategy takes then more time than the realization of a single activity. Therefore, the value assigned to the diagonal elements should be higher than other elements. In this thesis, the assumption is an arbitrary value of 0.8 for the diagonal elements and 0.1 to the other elements.

4.2.6 Fitness and Precision Metric

The discovered Map process models are supposed to reflect what really happened during a process enactment. An ideal discovered Map process model contains intentions and strategies coinciding with the frequent intentions and strategies as they really happened in a process.

Assuming that a given process would be observed *ad infinitum*:

- A *non-fitting* model is unable to characterize the process as it is actually enacted. This kind of model is not even able to reflect the traces used to estimate the model parameters.

- An *over-fitting* model lacks generalization, which means some traces do not fit into this kind of model.
- An *under-fitting* model suffer from the opposite problem, which leads to a lack of precision and allows behavior that would never occur.

Therefore, a good balance between over-fitting and under-fitting is of the utmost importance for Map process model discovery.

While there have been some research and progress in the process mining field to improve the quality of process models, a common framework to assess the quality of produced process model results is still lacking [Rozinat 2007]. However, the need for a framework that allows comparing the performance of the algorithms and evaluating the validity of the process model results seems fundamental.

This metric has the interesting property of taking into account both *Fitness* and *Precision* to optimize the Map process model, whereas classical metrics in process modeling address either fitness or precision (see [Rozinat 2007] for an overview of the existing metrics). The fitness refers to how many intentions and strategies in the traces are correctly captured (or can be reproduced) by the Map process model. The precision refers to how many more intentions and strategies are captured in the Map process model than was observed in the traces.

This metric will be detailed in Section 4.3.3, in which the question of *How to extract a Map process model that is not underfit, overfit, nor non-fit?* will be addressed.

4.3 The Proposed Method

4.3.1 Applying Hidden Markov Models

An HMM can be considered as the simplest Dynamic Bayesian network (DBN) [Murphy 2002]. The framework of HMMs in MMM context raises the following questions:

- Given a sequence of activities, how to estimate the transition and emission probabilities of the HMM model?
- What is the probability that the model of HMM generates a given activities sequence?
- What are the most likely strategies associated to a given activities sequence?

As explained in chapter 3, HMMs have proven to be appropriate for modeling the real world processes, particularly unobservable cognitive states [Hayashi 2003] such as underlying users' strategies. Beside, it turns out that the topology of HMMs is particularly adapted to model the relations between strategies and activities in the Map formalism. To make it clear, let us consider an example for a Map process model enacted with 2 strategies and an HMM realized with 2 hidden states (see

Figure 4.2). As shown in this figure, strategies are used to move from one intention to another and are made of one or several activities. For instance, the strategy 1 allows moving from intention a to intention b and it is made of activities a_1, a_3 and a_4 (the order is not important). The same structure can be found in an HMM, where hidden states generate observations. In other words, hidden state 1 generates the activities a_1, a_3 and a_4. This similar topology motivates using HMMs to model hidden states as strategies and the transitions between the strategies as intentions. The Section 4.3.3 will describe how to obtain the intentions from the transitions.

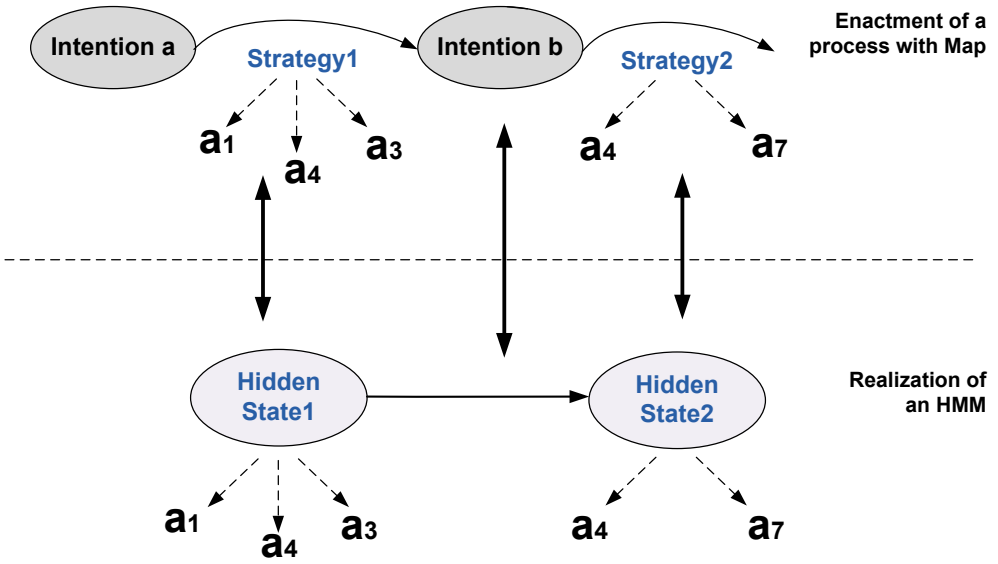

Figure 4.2: An example for a Map process model enacted with 2 strategies (above) and an HMM realized with 2 hidden states (below)

There are several algorithms that allow estimating the model parameters as explained in Section 4.3.2.

4.3.1.1 Mathematical Definition of HMMs

HMMs are stochastic Markov chains used for modeling a hidden sequence by a finite number of states. HMMs allow modeling the structure of complex temporal dependencies. In order to use HMMs, one needs to determine their topology and estimate their statistical parameters. The flexible nature of HMMs, due to their topological structure, allows defining dependencies between hidden states and observed data from the past to the future ones, according to the context.

HMM consists of two complementary Markov processes: hidden and observed processes. The states of hidden processes generate the symbol of observed processes. The states in hidden processes are not visible but the probability of being in a given state can be inferred by computing the Maximum Likelihood (ML) of symbols in

observed process. Thus, there are two probability parameters: probabilities of the symbol emission and the probabilities of the states transition.

Let X be a stationary and discrete source which generates a random observable sequence $(x_\ell, \cdots, x_\ell)$, of length ℓ. The generation process follows a Markov chain of order m (or with memory m), where m is finite, if the transition probability from one state to another depends on the m previous states. The choice of the order of Markov process (m) allows determining how far in the past one has to look to know the probability of the next state. The generation process can be written as follows:

$$\begin{aligned}\Pr(X_\ell = x_\ell | X_{(\ell-1)} = x_{(\ell-1)}, X_{(\ell-2)} = x_{(\ell-2)}, \cdots, X_1 = x_1) = \\ \Pr(X_\ell = x_\ell | X_{(\ell-1)} = x_{(\ell-1)}, X_{(\ell-2)} = x_{(\ell-2)}, \cdots, X_{(\ell-m)} = x_{(\ell-m)}) for \ell > m\end{aligned} \tag{4.1}$$

4.3.1.2 Hidden Markov Models Adapted to MMM

Hereafter, the formal definition of HMM and different learning approaches to compute their parameters are discussed.

On the one hand, in the Map metamodel, strategies are used to move from one intention to another. A strategy is made of one or several activities; consequently, intentions are not directly related to activities. Furthermore, activities are observable through event logs while strategies are hidden and must be inferred from observations. On the other hand, the topology of HMM permits modeling observed process (observations) in terms of hidden states (hidden part of observations). Therefore, to model the relationships between activities and strategies, HMM models activities as observed process and strategies as hidden states. Interestingly, this model also allows inferring intentions since once a strategy is inferred from activities, according to the Map formalism, a strategy leads to a target intention.

For this reason in this framework, the hidden states of HMM are modeled as users' strategies and the observed process as users' activities. The set of possible strategies is denoted as $\mathcal{S}$ and $\mathcal{A}$ as the set of possible activities.

For both Markov processes of an HMM, *i.e.*, hidden and observed processes, the topology (the order of the Markov process) is defined next.

4.3.1.3 Topology of HMM in MMM Framework

For each Markov process of an HMM, *i.e.*, for the hidden and the observed process, the topology (the order of process) must be defined. When the transition to the next state depends only on the current state, the Markov chain is of order 1 (model M_1). When the transition to the next state does not depend on any state, the Markov chain is of order 0 (model M_0). The M_1M_0 topology is chosen to model activities and related strategies. M_1 is chosen to model the users' strategies and M_0 is chosen to model the users' activities. This choice is justified by the fact that strategies are performed in a logical order by users; thus a strategy in step ℓ impacts the transition to the next strategy in the step $\ell + 1$. Even though the chosen model

for users' activities is M_0, an activity in a given step depends indirectly on the previous performed activity.

4.3.1.4 Hidden process: users' strategies

Let $\mathbf{s} = (s_1, \ldots, s_L) \in \mathcal{S}^L$ be a temporal sequence of users' strategies of length L. The topology M_1 is chosen for hidden process, which means that the strategy s_l at step l only depends on the strategy at step $\ell - 1$. A homogeneous Markov chain, which parameters are denoted by $\mathbf{T}$ and π, models the hidden process of strategies with:

$$\begin{aligned} \mathbf{T}(u, v) &= \Pr\left(s_\ell = v | s_{\ell-1} = u\right) \ \forall u, v \in \mathcal{S}, \ \ell \in [2, L], \\ \pi(u) &= \Pr\left(s_1 = u\right) \ \forall u \in \mathcal{S}. \end{aligned} \tag{4.2}$$

The vector π contains the probabilities of strategy at the initial state and the matrix $\mathbf{T}$ contains the transition probabilities for the following strategies, *i.e.*, the transition probabilities from any strategy at step $\ell - 1$ to any other strategy at step ℓ (including itself).

4.3.1.5 Observed process: users' activities

Let $\mathbf{a} = (a_1, \ldots, a_L) \in \mathcal{A}^L$ be the temporal sequence of users' activities of length L. The model M_0 is chosen for observed process, meaning that the emission of a_ℓ, at a given step ℓ, does not depend on any previous observation. It only depends on the hidden strategy at the same time step. The emission probability of an observation $a \in \mathcal{A}$ for a given strategy $u \in \mathcal{S}$ is given by:

$$\mathbf{E}(a, u) = \Pr\left(a | u\right). \tag{4.3}$$

The matrix $\mathbf{E}$ contains the emission probabilities of any activity for any strategy. Assuming that $\mathcal{S}$, $\mathcal{A}$ and π are known, the HMM model is fully described by $\mathcal{H} = \{\mathbf{E}, \mathbf{T}\}$, which represents the core information about the HMM behavior.

The transition probabilities are the probabilities of a hidden state at step ℓ to reach another hidden state at step $\ell + 1$ (or to stay in the same state). Let $N = |\mathcal{S}|$ and $M = |\mathcal{A}|$ be the cardinals of $\mathcal{S}$ and $\mathcal{A}$, *i.e.* the total number of hidden states and observed data, respectively. The following constraints must be verified by the parameters of an HMM:

$$\sum_{v=1}^{N} \mathbf{T}(u, v) = 1, \sum_{a=1}^{M} \mathbf{E}_u(a) = 1, \forall u \in \mathcal{S} \tag{4.4}$$

where

$$\mathbf{T}(u, v) \leqslant 1, \forall (u, v) \in \mathcal{S}; \mathbf{E}_u(a) \leqslant 1, \forall u \in \mathcal{S}, \forall a \in \mathcal{A} \tag{4.5}$$

From an initial hidden state given by π, an observation is generated according to $\mathbf{E}$, then and for each step of the process a new hidden state is generated according to $\mathbf{T}$ and a new observation is generated according to $\mathbf{E}$.

Figure 4.3 illustrates an example of the relationships between hidden states, observations as well as transition matrix $\mathbf{T}$, and emission matrix $\mathbf{E}$ in an HMM. For instance, $\mathbf{T}(State_3, State_2)$ represents the transition probability from $State_3$ to $State_2$ and $\mathbf{E}_3(a_2)$ represents the emission probability of observation a_2 in the state $State_3$. As mentioned earlier in MMM, the observations represent the users' activities and the hidden states represent users' strategies. Henceforth in this thesis, the *activities* and the *strategies* are used instead of observations and hidden states, respectively.

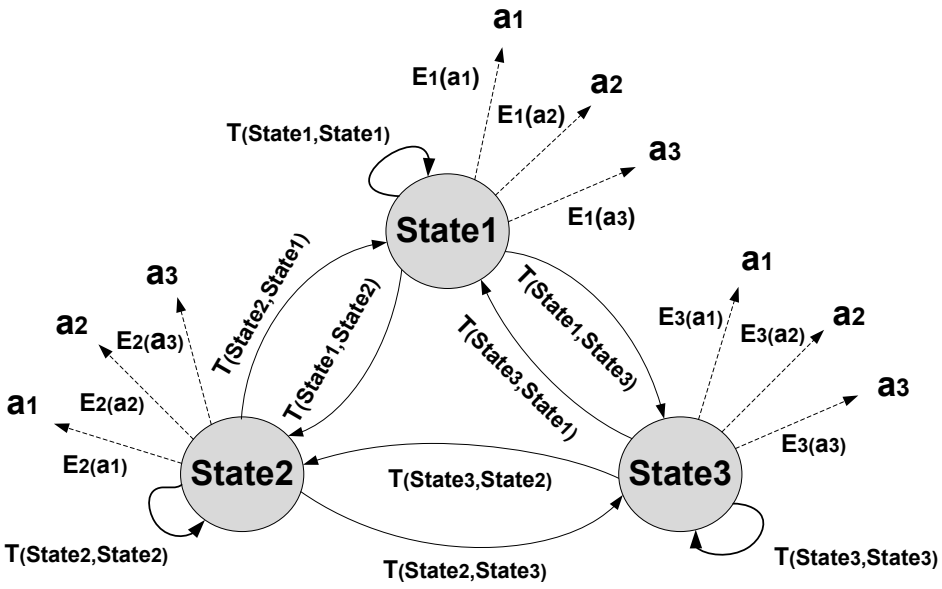

Figure 4.3: Example of the relationships between hidden states, observations, transition matrix $\mathbf{T}$, and emission matrix $\mathbf{E}$

4.3.2 Estimating Model Parameters

As discussed in Section 4.3.1.2, MMM highly relies on the emission matrix $\mathbf{E}$ and the transition matrix $\mathbf{T}$ to respectively characterize the occurrence of activities in each strategy and the transition probabilities from one strategy to another. For these reasons, it is extremely important that these two matrices precisely match the process under study. Therefore, it is essential to choose a learning approach to estimate the model parameters ($\mathcal{H} = \{\mathbf{T} \text{ and } \mathbf{E}\}$) which fit process model optimally. Estimating the parameters of an HMM depends on the learning approach.

As mentioned in Section 3.2.4, there are two learning approaches for estimating these matrices: *Supervised* or *Unsupervised* learning. These approaches, their conditions of use as well as their respective performances are discussed in sections 4.3.2.1 and 4.3.2.2. Figures 4.4 and 4.5 depict an overview of supervised and unsupervised learning.

4.3.2.1 Supervised Learning

Supervised learning aims at learning $\mathbf{E}$ and $\mathbf{T}$. If a sequence of activities $(a_1, \ldots, a_L)$ is available and the corresponding strategies are known, using the Maximum-Likelihood Estimation (MLE) [Myung 2003], this approach estimates the parameters $\mathbf{T}(u, v)$ and $\mathbf{E}_u(a)$ such that they analytically maximize the likelihood of having simultaneously the strategies $(s_1, \ldots, s_L)$ and the sequence of activities $(a_1, \ldots, a_L)$. Figure 4.4 depicts an overview of supervised learning.

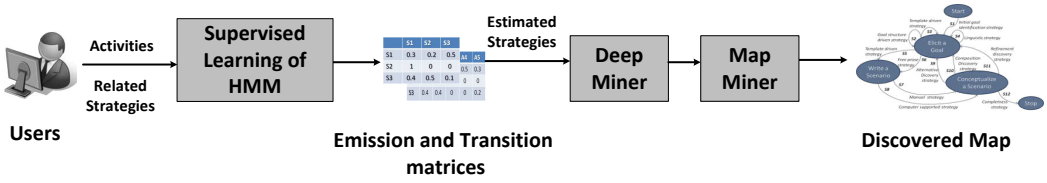

Figure 4.4: Overview of supervised Map Miner Method

Conditions of Use:

The conditions under which supervised learning can be used are very restrictive and the results might be biased. The application of this approach requires the knowledge of:

- The sets activities $\mathcal{A}$ and strategies $\mathcal{S}$,
- Some sequence of activities $(a_1, \ldots, a_L)$ and their associated sequences of strategies $(s_1, \ldots, s_L)$.

While the knowledge of $\mathcal{A}$ and $(a_1, \ldots, a_L)$ is generally not an issue (the possible activities of a given process are usually known and are recorded in traces), the knowledge of $\mathcal{S}$ and $(s_1, \ldots, s_L)$ is more problematic. Indeed, since strategies are the cognitive operators, the usual way to obtain the set $\mathcal{S}$ is to refer to experts. Since human judgment is involved, the obtained set $\mathcal{S}$ can be biased. The argument is that in a cognitive context such as a human's strategy and intention, it is impossible to properly label the training data, because this information is not observable. Moreover, human bias [Tversky 1974] is unavoidably introduced into training data labeling, which significantly impacts the learning process and may produce incorrect or uninformative process models. Moreover, strategies are usually not recorded in traces [Khodabandelou 2013]. Applying this learning method implies to conduct experiments specially designed to record traces of activities and traces of strategies. This condition highly restricts the range of use of this method in large-scale.

Performance:

Given N sequences of activities $(a_1, \ldots, a_L)$ and their associated N sequences of strategies $(s_1, \ldots, s_L)$, the aim of supervised learning is to find the couple $(\mathbf{E}^*, \mathbf{T}^*)$ which maximizes the likelihood of generating $(a_1, \ldots, a_L)$ and $(s_1, \ldots, s_L)$:

$$(\mathbf{E}^*, \mathbf{T}^*) = \arg\max_{\mathbf{E},\mathbf{T}} \prod_{n=1}^{N} \Pr(\mathbf{a}_n | \mathbf{s}_n, \mathbf{E}, \mathbf{T}) \tag{4.6}$$

Obtaining the coefficient of $\mathbf{T}^*$ amounts to counting the number of transitions from one strategy to another and obtaining the coefficients of $\mathbf{E}^*$ amounts to counting the number of occurrences of each activity during each strategy, as shown below:

$$\mathbf{T}^*(u, v) = \frac{\text{Num}(u, v)}{\sum_{w \in \mathcal{S}} \text{Num}(u, w)}, \ \forall (u, v) \in \mathcal{S}^2, \tag{4.7}$$

$$\mathbf{E}^*(u, a) = \frac{\text{Num}(a|u)}{\text{Num}(a)}, \ \forall u \in \mathcal{S}, \ \forall a \in \mathcal{A}, \tag{4.8}$$

where $\text{Num}(u, v)$ denotes the number of transitions from strategy u to strategy v in the traces $(\mathbf{s}_1, \ldots, \mathbf{s}_N)$, $\text{Num}(a)$ denotes the number of occurrences of activity a in $(\mathbf{a}_1, \ldots, \mathbf{a}_N)$ and $\text{Num}(a|u)$ denotes the number of occurrences of activity a while the strategy is u, in $(\mathbf{s}_1, \ldots, \mathbf{s}_N)$ and $(\mathbf{a}_1, \ldots, \mathbf{a}_N)$. The computation complexity of this method is very low since all the coefficients of $\mathbf{E}^*$ and $\mathbf{T}^*$ can be directly computed from the traces used for learning with (4.7) and (4.8).

The set of training sequences $(\mathbf{a}_1, \ldots, \mathbf{a}_N)$ and $(\mathbf{s}_1, \ldots, \mathbf{s}_N)$, is extremely important for the accuracy of the estimation of $\mathbf{E}^*$ and $\mathbf{T}^*$. If the set contains few traces, or they are not fully representative of all the traces that can be produced by the process, the HMM model learned out of it might suffer underfitting issues. From a practical point of view, this issue is common since the conditions to get usable training traces are complex (resulting in few usable traces).

4.3.2.2 Unsupervised Learning

Unsupervised learning estimates the matrices $\mathbf{E}$ and $\mathbf{T}$ based only on traces of activities. Since there is no prior knowledge on the strategies set $\mathcal{S}$, this method is significantly less biased than supervised learning but the associated computational complexity is high. Figure 4.5 depicts an overview of unsupervised learning.

The Baum-Welch algorithm (BWA) [Baum 1970] is the most commonly used algorithm in HMM framework to estimate the model parameters $\mathbf{E}$ and $\mathbf{T}$. Given N observed sequences of activities $(\mathbf{a}_1, \ldots, \mathbf{a}_N)$, the BWA finds the HMM parameters that locally maximize the probability of having these sequences generated by the HMM. More precisely, the BWA maximizes the likelihood of $\mathcal{H}$.

BWA makes use of the Expectation Maximization (EM) algorithm [Dempster 1977]. The aim of the EM algorithm is to estimate the MLE or *maximum a posteriori* of the statistical models (with latent variables)

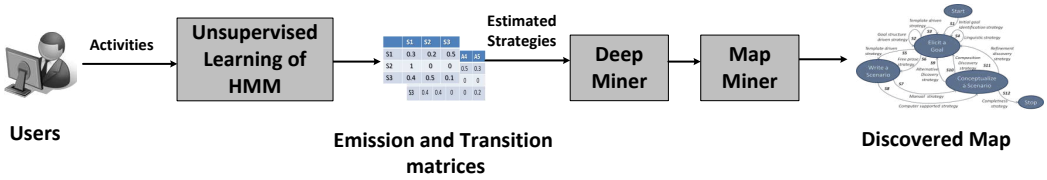

Figure 4.5: Overview of unsupervised Map Miner Method

parameters in an iterative way. The EM algorithm consists in two steps that alternate between an expectation (E-Step) and a maximization step (M-step).

The E-step consists in evaluating the log-likelihood of observations given the current HMM parameters. The M-step consists in updating the parameters of the HMM to increase the log-likelihood found in the E-step. The parameters found in the M-step are then used as the starting point of a new phase of E-step to determine the distribution of the latent variables. The iterations stop when the parameters of the HMM have converged. The E-step of the EM algorithm requires initial model parameters, arbitrarily chosen. Through this iterative procedure, it is proven that the BWA converges to a local optimum [Rabiner 1986]. This property ensures that the parameters found by the BWA are locally the parameters with the highest probability of generating the observed activities.

Conditions of Use:

For unsupervised learning, the required knowledge includes the set of activities $\mathcal{A}$, some traces of activities $(a_1, \ldots, a_L)$ and the cardinality of the set $|\mathcal{S}|$, *i.e.*, the number of possible strategies. Regarding strategies, neither the set $\mathcal{S}$ nor some traces of strategies $(s_1, \ldots, s_L)$ should be known, only the number of possible strategies is required. This parameter can be chosen by experts (*e.g*, as a way to set the level of complexity of the model) or can be set with techniques such as BIC [Burnham 2002] (see 4.3.2.4 for more details). Similarly to supervised learning, this choice introduces a bias, but given that only the number of strategies is set and not the strategies themselves, this bias is less important. The advantage of unsupervised learning is being applicable on traces comprising only activities traces.

Performance:

Given a trace made of N observed sequences of activities $(a_1, \ldots, a_L)$, the BWA finds the HMM matrices $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{T}}$ that locally maximize the probability of having these sequences generated by the HMM. More precisely, the BWA maximizes the

likelihood of $\mathbf{E}$ and $\mathbf{T}$:

$$\left(\tilde{\mathbf{E}}, \tilde{\mathbf{T}}\right) = \arg\max_{\mathbf{E},\mathbf{T}} \prod_{n=1}^{N} \Pr\left(\mathbf{a}_n | \mathbf{E}, \mathbf{T}\right) \tag{4.9}$$

As it is mentioned earlier, the number of strategies is required to know the dimensions of matrices $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{T}}$ since the BWA could not run without $\tilde{\mathbf{E}}$ and $\tilde{\mathbf{T}}$ being initialized.

What is interesting to note here is the fact the likelihood is not maximized depending on some traces of strategies $\mathbf{s}_1, \ldots, \mathbf{s}_N$, as it was the case for supervised learning. It means that the space in which the likelihood is maximized is larger than the space for supervised learning. As a consequence,

$$\max_{\mathbf{E},\mathbf{T}} \prod_{n=1}^{N} \Pr\left(\mathbf{a}_n | \mathbf{E}, \mathbf{T}\right) \geqslant \max_{\mathbf{E},\mathbf{T}} \prod_{n=1}^{N} \Pr\left(\mathbf{a}_n | \mathbf{s}_n, \mathbf{E}, \mathbf{T}\right). \tag{4.10}$$

In other words, the maximum likelihood of unsupervised learning is always higher than the maximum likelihood obtained by supervised learning since the latter comes from a constrained space. Unfortunately, the BWA cannot be guaranteed to converge to the global maximum likelihood since it is only proved to converge to a local optimum [Rabiner 1989]. The limit of convergence depends on the initialization of the matrices $\mathbf{T}$ and $\mathbf{E}$ and it is verified by experimental results (see Section 4.5), that a simple initialization of $\mathbf{T}$ and $\mathbf{E}$ leads to a maximum likelihood of unsupervised learning higher than supervised learning.

4.3.2.3 Summary of the Two Learning Approaches

In Table 4.5, a theoretical comparison of the two learning approaches is presented, based on the properties defined in Sections 4.3.2.1 and 4.3.2.2. Regarding the

	Traces for learning	**A-priori knowledge**	**Convergence speed (complexity)**	**Likelihood of the estimated parameters**
Supervised learning	Activities, Strategies	Set of activities, Set of strategies	Fast (one iteration)	Maximum over a restrained set
Unsupervised learning	Activities	Set of activities, Number of strategies	Slow (several iterations)	Local maximum

Table 4.5: Theoretical comparison of supervised and unsupervised learning

conditions of use, unsupervised learning can be applied on any trace comprising traces of activities, contrary to supervised learning which can be applied under more restrictive conditions. This makes unsupervised learning the most convenient method for practical use. However, since unsupervised learning is only proved to converge to a local maximum, it is not guaranteed to provide an estimated model with a better likelihood than supervised learning. In order to investigate this point, both approaches are compared on the same traces in Section 4.5.

4.3.2.4 Determining the Number of Strategies

The BWA requires the sets $\mathcal{A}$ and $\mathcal{S}$ to be known or at least, their cardinality, *i.e.* $|\mathcal{S}|$ and $|\mathcal{A}|$ for the algorithm to run. Regarding the set of activities $\mathcal{A}$, it can simply be obtained by identifying the different activities in the trace. However, obtaining the set of strategies $\mathcal{S}$ is impossible since there is no information about strategies in the trace. There are three ways to obtain the number of strategies:

- This parameter can be chosen by experts. This is interesting since it allows setting the level of complexity of the model, to meet some *a priori* expectations of the model. However, as this choice involves human intervention, it introduces a bias.

- Several criteria exist to determine the number of hidden states, such as Bayesian Information Criterion (BIC) [Burnham 2002]. This metric allows the comparison of HMM models with different numbers of hidden states, trained on the same underlying data. BIC penalizes the likelihood of the model by a complexity factor proportional to the number of parameters θ in the model and the number of training observations $\mathcal{R}$:

$$BIC = -2\log\left(\Pr(\mathcal{A}|\mathcal{H})\right) + \theta\log\left(\mathcal{R}\right), \tag{4.11}$$

  where $\theta = (J^2 + J \times F)$, and J^2 and $J \times F$ represent the number of parameters in transition matrix and emission matrix, respectively.

  Although BIC can ensure a result for every sequence of activities, this trade-off does not allow generating the model with the best likelihood when the model has a high complexity factor.

- The method that is used in the thesis (see Section 4.5.2) to set the right number of strategies is heuristic. It consists in generating several HMM models with different numbers of strategies and observing the associated emission matrices. It occurs that as the number of possible strategies increases, the number of different strategies obtained in the emission matrices reaches a threshold. It means that when the number of possible strategies is too high, the BWA produces an emission matrix with several identical strategies. Consequently, to set the right number of strategies of the model, this observed

threshold is chosen. This method has the advantage of being adaptable for different traces. The drawback of this method is its computation complexity.

4.3.3 Developing Deep Miner Algorithm

Once the strategies are estimated, a Map process model can be extracted from these matrices generated by the BWA. This can be realized by Deep Miner algorithm that especially developed to construct the sections of Map process model out of the transition matrix. However, the question of measuring the quality of this Map remains. We recall that the matrices generated by the BWA are:

- An emission matrix $\mathbf{E}$, giving the probabilities of generating any activity while performing a strategy. In other words, for any strategy $s \in \mathcal{S}$, it gives the activities the strategy is composed of.
- A transition matrix $\mathbf{T}$, giving the probabilities of transition between any couple of strategies $(s, s') \in \mathcal{S}^2$.

Clearly, there is a strong link between the transition matrix and the topology of the Map process model to be extracted. However, in the general case, it is a difficult task to manually extract a Map from a transition matrix while verifying the two following constraints: (i) any transition between possible strategies in the transition matrix should be possible on the Map, (ii) any transition between possible strategies in the Map should be possible in the transition matrix.

The first constraint can be seen as a criterion for fitness since it ensures that all the transitions learned from the trace are present in the Map. The second constraint corresponds to a criterion of precision since it aims at avoiding introducing extra-transitions in the Map that are not learned from the trace. Our goal is to find the Map that best satisfies both of them. Additionally, transitions probabilities in $\mathbf{T}$ have different values according to their respective importance for the model. This means that some transitions are more important than others and in consequence, the fitness and the precision are also more important for these transitions. In the next part, a metric is defined which is a trade-off between fitness and precision and also captures the relative importance of transitions.

4.3.3.1 Proposed Metrics of Fitness and Precision

The topology of a Map $\mathbf{m}$ can be defined by the set of its sections, each one comprising a source sub-intention, a strategy and a target sub-intention. It can be formally written as:

$$\mathbf{m} = (m_k)_{k \in \{1,\dots,K\}}, \tag{4.12}$$

where k denotes the index of a section and K is the total number of sections of the Map. For each $k \in \{1, \dots, K\}$, $m_k = (i, j, s) \in \mathcal{I} \times \mathcal{S} \times \mathcal{I}$. The component $m_k(1)$ is the source sub-intention of section k, $m_k(2)$ is the target sub-intention, and $m_k(3)$ is the strategy of section k. On the Map $\mathbf{m}$, a transition from strategy s to strategy

s' is possible if and only if there exist $(k, k') \in \{1, \ldots, K\}^2$ such that $m_k(3) = s$, $m_{k'}(3) = s'$, and $m_k(2) = m_{k'}(1)$. In the following, the symbol α is used to denote if a transition is possible or not in the Map:

$$\alpha_{s,s'} = \begin{cases} 1 \text{ if } \exists (k, k') \in \{1, \ldots, K\}^2 \text{ such that } m_k(3) = s, \\ m_{k'}(3) = s', \text{ and } m_k(2) = m_{k'}(1), \\ 0 \text{ otherwise.} \end{cases} \tag{4.13}$$

In the transition matrix $\mathbf{T}$, the only valid transitions is considered with a probability above a given threshold ε. The value of ε has to be chosen heuristically, to counter the effects of noise and artifacts in the trace. Figure 4.6 depicts an example showing the effect of threshold ($\varepsilon = 0.2$) before and after applying it on the transition matrix. As shown on the right of the figure, all the transitions lower than 0.2 are eliminated. Therefore, according to the value of ε, Deep Miner takes into account certain transitions.

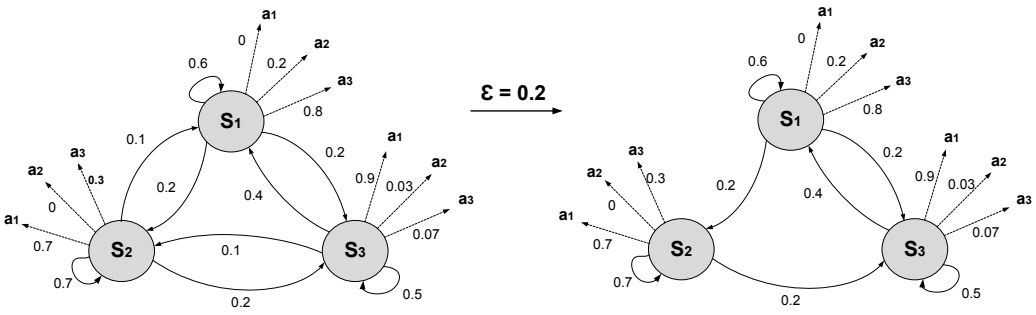

Figure 4.6: An example showing the effect of threshold on the transitions

This can be defined as follows:

$$\omega_{s,s'} = \begin{cases} 1 \text{ if } \mathbf{T}(s, s') \geqslant \varepsilon, \\ 0 \text{ if } \mathbf{T}(s, s') < \varepsilon. \end{cases} \tag{4.14}$$

Classically, the criteria of fitness and precision between $\mathbf{T}$ and $\mathbf{m}$ can be expressed by the expressions known as recall and precision. In our context, these two expressions are defined as :

$$\mathrm{Rec}(\mathbf{T}, \mathbf{m}) = \frac{\sum_{s,s'} \omega_{s,s'} \alpha_{s,s'}}{\sum_{s,s'} \omega_{s,s'}}, \tag{4.15}$$

$$\mathrm{Pre}(\mathbf{T}, \mathbf{m}) = \frac{\sum_{s,s'} \omega_{s,s'} \alpha_{s,s'}}{\sum_{s,s'} \alpha_{s,s'}}. \tag{4.16}$$

The numerator of both expressions is the number of significant transitions in $\mathbf{T}$ that are present on the Map $\mathbf{m}$, while the denominators are the number of significant transitions in $\mathbf{T}$ and the number of transitions on $\mathbf{m}$, respectively.

Since our goal is to find a Map that fits best the transition matrix with respect to both recall and precision, the classical F-measure can be used which expression is:

$$F_1(\mathbf{T}, \mathbf{m}) = 2\frac{\text{Pre}(\mathbf{T}, \mathbf{m})\text{Rec}(\mathbf{T}, \mathbf{m})}{\text{Pre}(\mathbf{T}, \mathbf{m}) + \text{Rec}(\mathbf{T}, \mathbf{m})}. \tag{4.17}$$

4.3.3.2 Optimization problem

Once the proper metric has been defined, a Map that maximizes this metric must be found. The solution of this problem belongs to the set

$$\mathcal{M} = \arg\max_{\mathbf{m}} F_1(\mathbf{T}, \mathbf{m}). \tag{4.18}$$

Since the goal is obtaining a Map with the simplest structure, the solution with the lowest number of sections is chosen. In other words, the solution is

$$\mathbf{m}^* = \arg_{\mathbf{m}\in\mathcal{M}} \min |\mathbf{m}|, \tag{4.19}$$

where $|\mathbf{m}|$ stands for the number of sections in $\mathbf{m}$. However, finding $\mathbf{m}^*$ is a difficult task since $\mathbf{m}$ generally belongs to a high-dimension space. Indeed, it can be shown that there are $2^{|\mathcal{S}|^2}$ possible Maps for $|\mathcal{S}|$ different strategies. Consequently, computing all the possible Maps with a brute force [1] method then comparing their F-measures is not an option. Instead, an algorithm is developed that solves (4.19) with a complexity bounded by $|\mathcal{S}| * (|\mathcal{S}| - 1)$. This algorithm is detailed below.

Algorithm 1 How to obtain a Map from $\mathcal{S}$, $\mathbf{T}$, and ε.

Data: strategy set $\mathcal{S}$, transition matrix $\mathbf{T}$, threshold ε
Result: Pseudo-Map $\mathbf{m}^*$
for *each strategy* $s \in \mathcal{S}$ **do**
  associate to s a target sub-intention i_s
end
for *each strategy* $s \in \mathcal{S}$ **do**
  for *each strategy* $s' \in \mathcal{S},\ s' \neq s$ **do**
    if $\mathbf{T}(s, s') \geqslant \varepsilon$ **then**
      create a section from i_s to $i_{s'}$ with strategy s'
    end
  end
end

The first part of algorithm 1 associates a target sub-intention to each strategy of $\mathcal{S}$. In the second part, if a transition probability from strategy s to strategy s' is above the threshold ε, a section is added to the Map from the target sub-intention

[1] Brut force is a method of mathematical proof in which the statement to be proved is split into a finite number of cases and each case is checked to see if the proposition in question holds [Inglis 2011]

of s to the target sub-intention of s'. This section ensures that the transition given by $\mathbf{T}$ is also present in the Map. With this algorithm, recall and precision, defined in (4.15) and (4.16), have the advantage of being equal to 1. Indeed, ε defines the granularity of the Map. When ε is close to 0, almost all the transitions from the unsupervised model are present in the obtained Map. Consequently, the likelihood of the obtained Map is high but the Map is hardly understandable by humans since it has too many sections. However when ε increases, the number of sections, as well as the likelihood of the obtained Map, decrease. The Map gets more easily understandable by humans but it is not as accurate in terms of transition. Figure 4.7 depicts an overview of the Deep Miner algorithm. Its input is estimated strategies and output is a pseudo-Map.

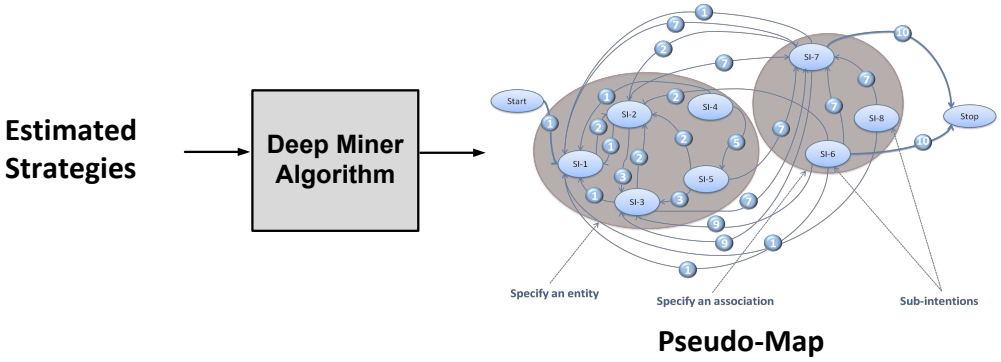

Figure 4.7: Overview of Deep Miner Algorithm

4.3.3.3 An Example for the Construction of a Map

Let us consider a transition matrix $\mathbf{T}$ as follows:

$$\mathbf{T} = \begin{pmatrix} 0.6 & 0 & 0.2 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 & 0 & 0.4 \\ 0 & 0 & 0.6 & 0.4 & 0 \\ 0 & 0.1 & 0 & 0.2 & 0.7 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Each row represents a strategy: the first row corresponds to S_1, the second row corresponds to S_2, and so forth. This procedure is the same for the column: the first column refers to S_1, the second column refers to S_2, and so forth. For example, the third element of the first row of the matrix expresses the transition from S_1 to S_3. The probability of this transition is 0.2 as mentioned in the matrix. The second element of the first row, for instance, expresses the transition from S_1 to S_2. The probability of this transition is equal to 0, which means there is no transition from

S_1 to S_2. On the contrary, there is a transition from S_2 to S_1 (the first element of the second row) since the probability of this transition is not equal to zero (it is 0.2).

The first step of the construction of a Map from this transition matrix consists in eliminating all the diagonal values, which means the couples (S_1,S_1), (S_2,S_2), (S_3,S_3), (S_4,S_4), and (S_5,S_5). Indeed, the diagonal values indicate the time needed to execute a strategy. For example, the transition (S_1,S_1) represents the time for executing the strategy S_1. These strategies are then not interesting for the construction of a Map, as only the transition between two different strategies are interesting. Therefore, all these values must be deleted from the matrix. Note that after each elimination the values must be normalized. The left values must be divided by the sum of the left values of the row. The modified matrix is shown below:

$$\mathbf{T} = \begin{pmatrix} 0 & 0 & 0.2/0.4 & 0.2/0.4 & 0 \\ 0.2/0.8 & 0 & 0.2/0.8 & 0 & 0.4/0.8 \\ 0 & 0 & 0 & 0.4/0.4 & 0 \\ 0 & 0.1/0.8 & 0 & 0 & 0.7/0.8 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The second stage consists in eliminating the elements of this matrix which are smaller than the threshold (ε). For instance, if this threshold is adjusted to 0.2, all the elements smaller than 0.2 must be eliminated. Note that each row should be normalized after an elimination. In the above mentioned transition matrix, only one element is smaller than 0.2 (the second element of the fourth row).

$$\mathbf{T} = \begin{pmatrix} 0 & 0 & 0.5 & 0.5 & 0 \\ 0.25 & 0 & 0.25 & 0 & 0.5 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Third stage consists in assigning to each strategy existing in the matrix a target sub-intention. In other words, this stage consists in constructing a couple, <Strategy,Target sub-Intention>. Indeed, according to the Map formalism each strategy leads to an intention. This definition can be extended for the sub-intentions. Therefore, each estimated strategy in the matrix must lead to a target sub-intention. This procedure is illustrated in Figure 4.8. As shown in this figure the five strategies are related to a target sub-intention index, for each row/column of the matrix. For readability reasons, the sub-intention are labeled with the same index of its related strategy.

Fourth stage consists in constructing the relationships between strategies according to the transition matrix. To do so, the starting point is the first row. As

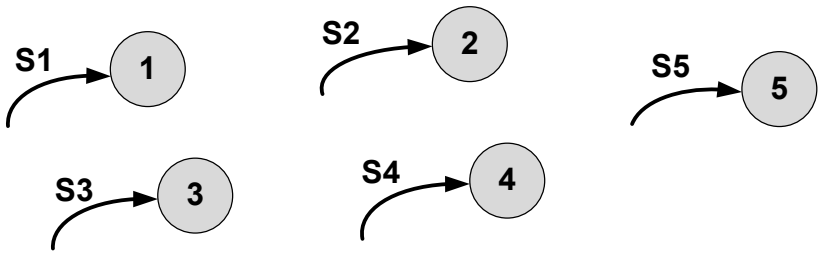

Figure 4.8: Assigning intention labels to each existing strategy

mentioned earlier this row corresponds to strategy S_1. This strategy is followed by the strategies S_3 and S_4 since there are transition probabilities between these strategies (0.5 for both of them). Therefore, S_1 is followed by S_3 or S_4. Figure 4.9 depicts the construction of the second part of the Map.

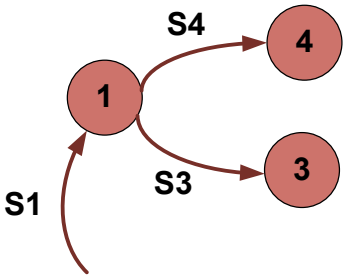

Figure 4.9: Relating strategy S_1 to strategies S_3 and S_4

The second row of the transition matrix corresponds to strategy S_2. This strategy is followed by the strategies S_1, S_3, and S_5. Therefore, strategy S_2 should be connected to S_1, S_3, and S_5. Figure 4.10 depicts the construction of this part of the Map.

The third row of the transition matrix corresponds to strategy S_3. This strategy is followed by strategy S_4. Therefore, strategy S_3 should be linked to strategy S_4. Figure 4.11 depicts the construction of this part of Map.

The fourth row of the transition matrix corresponds to strategy S_4. This strategy is followed by strategy S_5. Once again there is only one arrow that goes from S_4 to strategy S_5. Figure 4.12 depicts the construction of this part of Map.

The final stage consists in determining **Start** and **Stop** intentions. According to the Map formalism, the intention **Start** corresponds to the beginning of the process; consequently there is no incoming transition that goes in it. According to this formalism **Stop** intention corresponds to the end of the process; consequently there is no outgoing transition that goes out of it. Respecting these definitions,

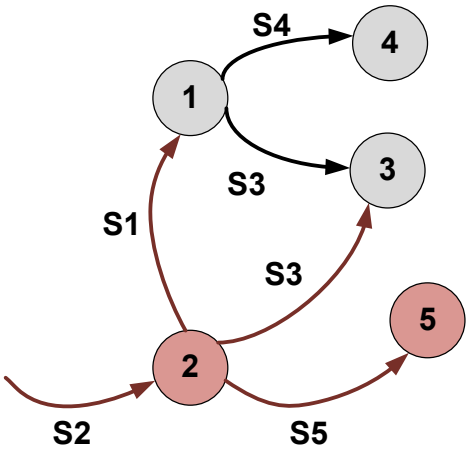

Figure 4.10: Relating strategy S_2 to strategies S_1, S_3, and S_5

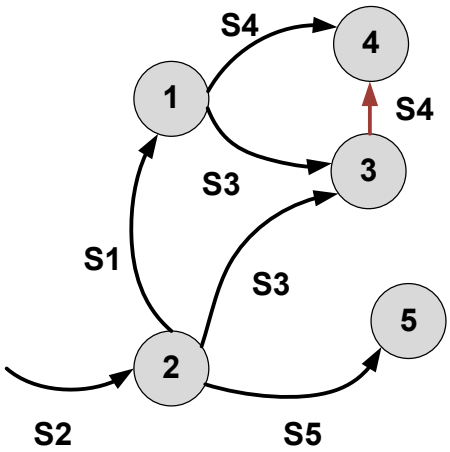

Figure 4.11: Relating strategy S_3 to strategies S_4

to determine **Start** intention, we should look for the strategies with no incoming strategies. The strategy S_2 respects this definition. To determine the **Stop** intention, we should look for the strategies with no outgoing strategies. The strategy S_5 respects this definition. Figure 4.13 illustrates the Map process model with the **Start** and **Stop** intentions.

As shown here, the Deep Miner algorithm constructs the sections of a Map from the estimated transition matrix. These sections lead to the sub-intentions. To construct a Map process model with high-level intentions respecting the sections defined with Deep Miner algorithm, Map Mainer is proposed.

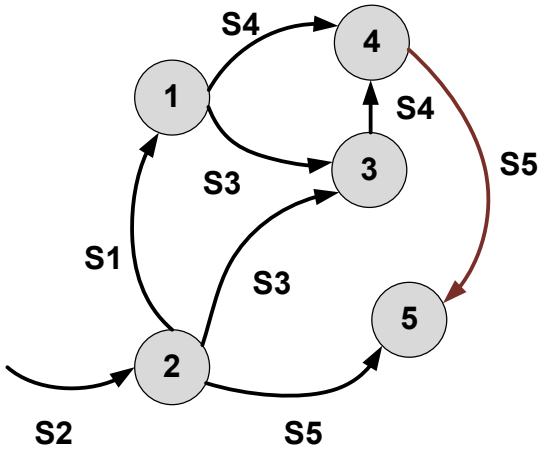

Figure 4.12: Relating strategy S_4 to strategies S_5

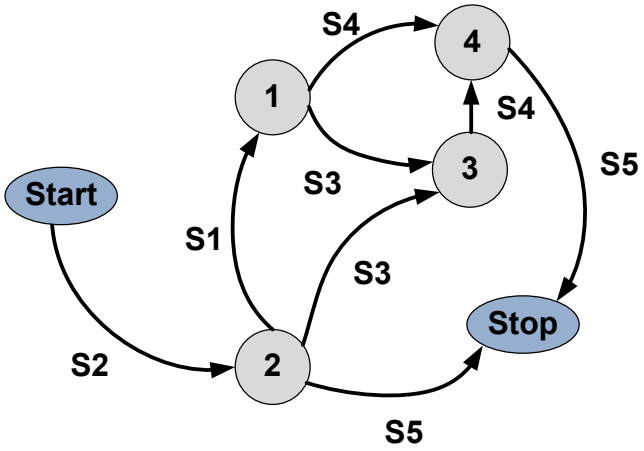

Figure 4.13: Determining **Start** and **Stop** intentions

4.3.4 Developing Map Miner Algorithm

Granularity refers to the level of detail of a process model. While a Map process model with a *coarse-grained* granularity represents high-level of abstraction for intentions (*e.g.*, organizational intentions), a Map process model with a *fine-grained* granularity, called a *pseudo-Map*, provides low-level of abstraction for intentions, called sub-intentions. Depending on the situation at hand, one can define the nature of granularity that is needed. This affects the kind of guidance and explanation that the model can provide [Rolland 1998a].

For instance, project managers or middle managers require rather coarse-grained

process description as they want to gain an overview of time, budget, and resource planning for their decisions. In contrast, software engineers, users, testers, analysts, or software system architects will prefer a fine-grained process model where the details of the model can provide them with instructions and important execution dependencies such as the dependencies between people. Hybrid formalisms such as Process Weaver [Fernstrom 1991] uses different notations for coarse-grained and fine-grained aspects of process.

4.3.4.1 Determining the Level of Abstraction for the Intentions

Given that the pseudo-Map obtained from Deep Miner algorithm has a high degree of granularity (in terms of sub-intentions and sections), it can be useful to extract some higher-level Maps of the same process from this pseudo-Map. An algorithm has been developed to automatically perform this task and is presented in this section. It falls into three main parts:

1. The sub-intentions from the pseudo-Map are represented in a space in which they can be classified into clusters.
2. A clustering algorithm, namely *K-means*, is applied on the sub-intentions in order to group them into clusters of intentions. This allows determining the level of abstraction for the intentions. Note that the number of intentions is a parameter that has to be chosen. The choice of this parameter allows researchers obtaining Maps with different level of granularity.
3. Finally, a Map process model is rebuilt from the new groups of intentions with updated sections.

Figure 4.14 represents an overview of Map Miner Algorithm.

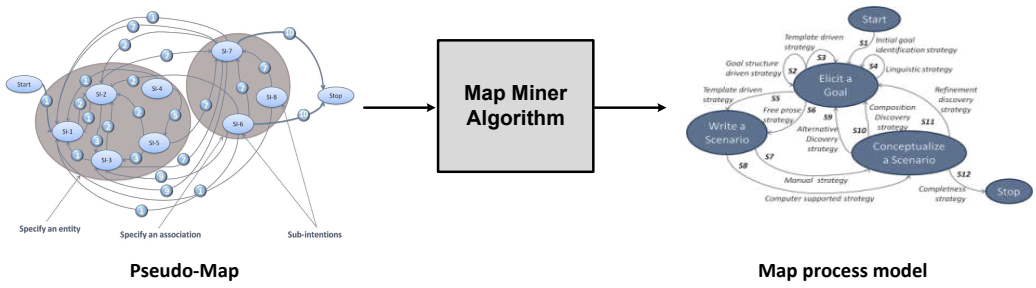

Figure 4.14: Overview of Map Miner Algorithm.

4.3.4.2 Sub-intentions Representation in the Space

Before clustering sub-intentions into groups of intentions, one needs to represent each sub-intention in a space that trustfully accounts for the topology around the

sub-intention in the pseudo-Map. Given that each sub-intention is connected to other sub-intentions by strategies, a proper way to represent sub-intentions is to indicate to which other sub-intentions it is connected. Since the Map is an oriented graph, a difference is made between sub-intentions from which there is a strategy going to the sub-intention to be represented and sub-intentions that can be fulfilled with strategies from the sub-intention to be represented.

From a formal perspective, let us consider a pseudo-Map with N sub-intentions. The sub-intention $i_n \in \mathcal{I}$ is represented by a vector v_n in a space of dimension $2N$ such that the first N coefficients correspond to the sub-intentions from which there is a strategy going to i_n, and the final N coefficients correspond to the sub-intentions that can be fulfilled from i_n.

- For all $n' \in [1, N]$, if there is a section from $i_{n'}$ to i_n then $v_n(n') = 1$, otherwise $v_n(n') = 0$.

- For all $n' \in [1, N]$, if there is a section from i_n to $i_{n'}$ then $v_n(2n') = 1$, otherwise $v_n(2n') = 0$.

- Since i_n is implicitly considered to be connected to itself, then $v_n(n) = 1$ and $v_n(2n) = 1$.

Let us consider a fragment of a pseudo-Map with 8 sub-intentions (Figure 4.15). The sub-intentions 'SI-1', 'SI-2', $\cdots$,'SI-8' are respectively represented by the vectors v_1, v_2,$\cdots$, v_8 in the space. For instance, the vectors v_3 represents 'SI-3'.

$v_3 = (01110000 \quad 00100110)$

This vectors is composed of the values '0' or '1'. The vector size is equal to 16; this size is due to possibilities of 8 incoming sub-intentions SI-1, SI-2, $\cdots$, SI-8 and 8 outgoing sub-intentions SI-1, SI-2, $\cdots$, SI-8 (comprising the sub-intention it-self). The left side of the vector defines the incoming sub-intentions and the right side defines the outgoing sub-intentions. For instance, in v_3 the incoming sub-intentions are SI-2 and SI-4 and the outgoing sub-intentions are SI-6 and SI-7. Therefore, in v_3 the value of '1' is assigned to the incoming sub-intentions SI-2 and SI-4 (at the left side) and the outgoing sub-intentions SI-6 and SI-7 (at the right side). The value '0' is assigned to the other sub-intentions.

With this representation, two sub-intentions connected to similar sub-intentions will be represented with a short distance between them, while two sub-intentions connected to different sub-intentions will be represented with an important distance between them. This way, the clustering algorithm that is applied on the sub-intentions can group sub-intentions efficiently.

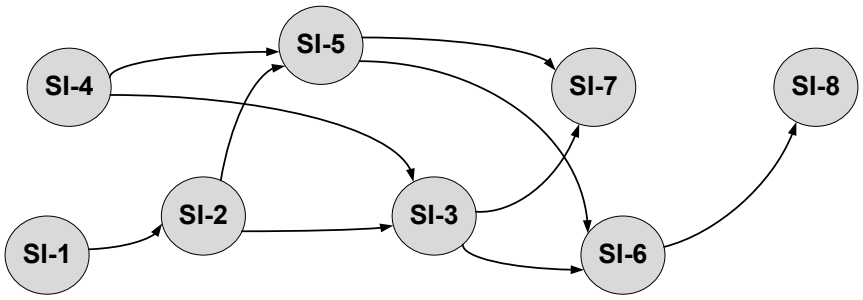

Figure 4.15: A fragment of a pseudo-Map with 8 sub-intentions

4.3.4.3 Clustering sub-intentions into high-level intentions

Once the sub-intentions are represented in the space described in Section 4.3.4.2, a clustering algorithm can be applied to group them into clusters of intentions. In this work, the *K-means* [Hartigan 1979] algorithm is applied to perform this task. This algorithm works as the following way: given a number K of clusters and a set of points $(v_n)_{n\in[1,N]}$, the algorithm determines the gravity center c_k of each cluster k, such that the sum of the distances from the points to the center of their cluster is minimized. In other words, it minimizes the sum,

$$\sum_{1:N} d\left(v_n, c(v_n)\right), \tag{4.20}$$

where $c(v_n)$ is the center of gravity which is the closest to v_n. For example, if v_n is closer to c_k, $c(v_n) = c_k$. And $d(.,.)$ is a distance between two points.

Said in another way, the *K-means* algorithm finds K groups of sub-intentions such that in each group, sub-intentions are in a same area of the Map process model. The mapping between the sub-intentions and the intentions are called g , such that $g(n)$ is the intention of sub-intention n.

4.3.4.4 Rebuilding the Map

To obtain a new Map from the clusters of intentions, all the previous sub-intentions are replaced by the intention of their group. The sections also need to be updated to take into account the simplified topology of the Map. We recall that the pseudo-Map discovered by the Deep Miner algorithm is denoted by $\mathbf{m}^*$. Note that the identical sections have to be removed from the discovered Map. Algorithm 2 shows how to rebuild a Map process model from K clusters of intentions given that the mapping from sub-intentions to intentions obtained from K-means is denoted by g.

Algorithm 2 Rebuilding a Map from K clusters of intentions.

Data: Map $\mathbf{m}^*$, mapping g
Result: Map $\widetilde{\mathbf{m}}$
for *each section* m_u^*, $u \in [1, U]$ **do**
  $\widetilde{m}_u(1) := g\left(m_u^*(1)\right)$
  $\widetilde{m}_u(2) := g\left(m_u^*(2)\right)$
end
Remove identical sections in $\widetilde{\mathbf{m}}$

4.4 Method for the Discovery of Map Path

Once the parameters of HMMs (matrices **E** and **T**) are estimated, it is possible to find the most likely strategies related to a given sequence of activities through Viterbi Algorithm (VA) [Forney Jr 1973] (see Figure 4.16). This means identifying the path (*i.e.*, a sequence of strategies) fulfilled by users.

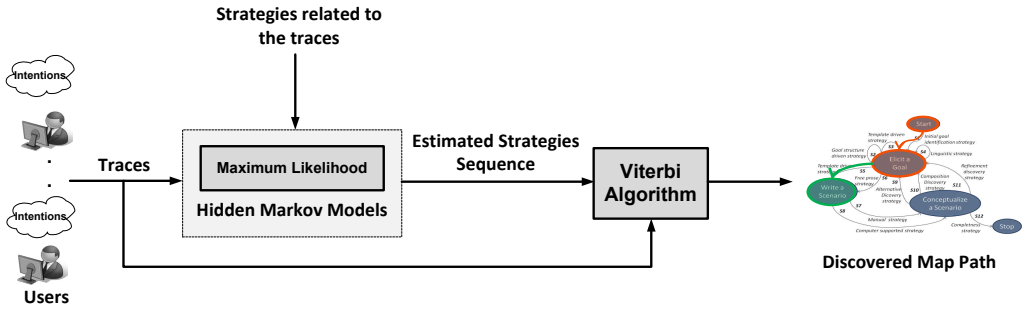

Figure 4.16: Overview of Map path discovery

This section describes VA from a theoretical point of view and Section 4.6 presents its application on the example.

Given a sequence of activity $\mathcal{A}_{1:N}$ of length N, one could generate all the possible strategies of length N. Then, for each strategy $\mathcal{S}_{1:N}$, one could compute the probability $\Pr\left(\mathcal{A}_{1:N}|\mathcal{S}_{1:N}\right)$. However, this is a brute-force search and it cannot be used to compare all the possible strategies for complexity reasons. For instance, if the number of strategies is $\mathcal{C}$, the complexity will be $\mathcal{C}^N$, which increases exponentially with N.

Instead, the VA is used to obtain, from a given observed sequence, the most likely hidden sequence of strategies that might generate it. The VA is commonly used in the context of HMM; this algorithm is able to calculate the probability that an observation has been changed into another, and radically simplifies the complexity of the search for the most likely hidden sequence. Thereby, the exponential complexity becomes linear.

To use the VA, it is necessary to know the estimated model parameters, **E** and

T. Note that a given sequence of activities, with length N, may be generated by many related strategies with the same length; nevertheless, one sequence among all has the highest probability of emergence. In other words, this sequence is the most likely sequence of strategies, which generates the related sequence of activities. The mathematical description of this phase is presented hereafter:

Given a sequence of activities $\mathcal{A}_{1:N}$, the estimated parameters $\hat{\mathbf{E}}_u(a)$ and $\hat{\mathbf{T}}(u,v)$ and the initial probabilities for each strategy, let $\delta_N(S)$ be the highest probability path through all hidden states. The VA tries to find the hidden associated sequence of strategies $\bar{\mathcal{S}}_{1:N}$ which maximizes:

$$\Pr\left(\mathcal{S}_{1:N}|\mathcal{A}_{1:N}\right) \tag{4.21}$$

The problem can also be written as:

$$\delta_{\mathbf{T}}\left(\mathcal{S}\right) = \arg\max_{1:N} \Pr\left(\mathcal{S}_{1:N}|\mathcal{A}_{1:N}\right) \tag{4.22}$$

iteratively, and then uses a backtracking process to decode the sequence of hidden states taken along the path of maximum likelihood. VA enables to account for the history of the users' activities.

4.5 Method Exemplification

To evaluate the proposed method, as well as comparing the supervised and unsupervised learning, MMM is applied on the E/R diagrams traces example, introduced in Section 4.1. We recall that, according to the prescribed Map (Figure 4.17), students can select ten strategies (represented as edges of the prescribed Map) to fulfill four intentions, **Start** (I_1), **Specify an entity** (I_2), **Specify an association** (I_3) and **Stop** (I_4) (represented as nodes on the prescribed Map). To select a strategy, students have to carry out activities. There are 12 activities related to the prescribed Map. Table 4.6 gives the names of all the activities and Table 4.7 shows in detail the link between strategies and related activities in the prescribed Map. Note that, since the students should follow the instructions, the prescribed Map process model may introduce a bias regarding the results that will be obtained by applying MMM on the students' traces.

In Section 4.5.3 the results of the two learning approaches for the example are compared in terms of human effort requirement, convergence speed, computation complexity, and likelihood.

4.5.1 MMM Using Supervised Learning

4.5.1.1 Estimating Model Parameters

First, supervised learning is applied on activities traces to obtain the model parameters defined in Section 4.3.2. Note that the traces have to be compatible with supervised learning approach. Indeed, they have to comprise traces of activities and corresponding traces of strategies. Since we set up the experiment, we were

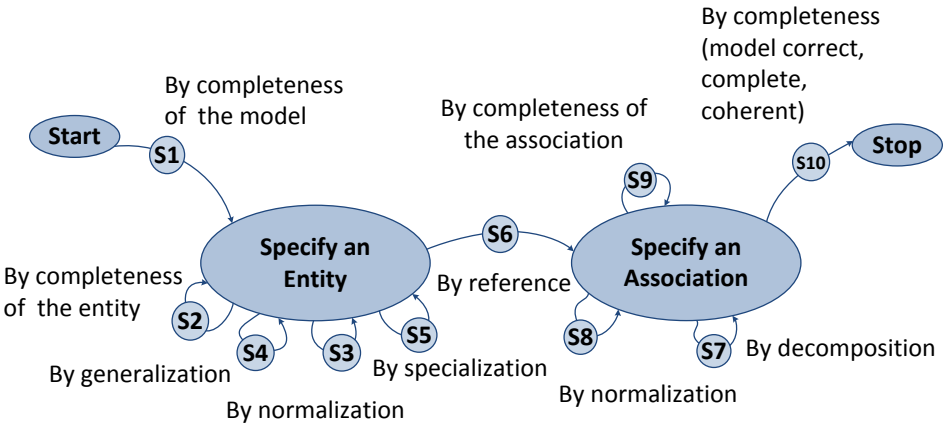

Figure 4.17: Prescribed Map Process Model for construction of E/R diagrams

Index	Related activities
a_1	Create an entity
a_2	Link attribute to entity
a_3	Create an entity Create generalization link
a_4	Create an entity Create specialization link
a_5	Delete an entity
a_6	Delete link attribute to entity
a_7	Define primary key
a_8	Delete link attribute to entity Create an entity Link association to entity
a_9	Create an association Link association to entity
a_{10}	Delete an association Delete link attribute to association
a_{11}	Link attribute to association
a_{12}	Check the model

Table 4.6: Activities and their labels for E/R diagrams

able to record students' strategies sequences in addition to their activity sequences. This procedure is so-called labeling process. This allows estimating the model parameters by supervised learning. After modeling the activities traces by HMM and

Index	Name of strategies	Related activities (activities index)
S_1	By completeness (model)	Create entity (a_1)
S_2	By completeness (entity)	Link attribute to entity (a_2)
S_3	By normalization	Delete Link attribute to entity (a_6), Delete entity (a_5), Define primary key (a_7)
S_4	By generalization	Create entity (a_1), Create generalization link (a_3)
S_5	By specialization	Create entity (a_1), Create specialization link (a_4)
S_6	By reference	Delete link attribute to entity, Create entity, Create association, Link association to entity (a_8), Create association, Link association to entity (a_9)
S_7	By decomposition	{Create association, Link association to entity} (a_9)
S_8	By normalization	{Delete association, Delete Link attribute to association}(a_{10})
S_9	By completeness (assoc.)	Link attribute to association (a_{11})
S_{10}	By completeness (final)	Check the model (a_{12})

Table 4.7: Map strategies and related activities of the example

applying supervised learning, the model parameters are estimated.

4.5.1.2 Applying Deep Miner Algorithm

Deep Miner algorithm using the transition matrix **T** thereby computed, generates a pseudo-Map (Figure 4.18) with a threshold adjusted to $\varepsilon = 0.05$. This threshold is obtained by heuristics. Generally, this threshold is an arbitrary value and can be determined by the subject-matter expert.

4.5.1.3 Applying Map Miner Algorithm

The *sub-intentions* are denoted by $\{\text{SI-1}, \cdots, \text{SI-9}\}$. Map miner algorithm groups them into 4 high-level intentions. This choice for the number of intentions allows comparing the discovered Map with the prescribed Map in the same level. Figure 4.19 provides the discovered Map at the same granularity level as the prescribed Map. Since the assumption of supervised learning is that for a given activity, the related strategy and intention are known. Therefore, the names of strategies and the intentions of the discovered Map are the same as the prescribed Map.

In Figure 4.19, the strategies discovered by Map Miner algorithm that match the prescribed Map sections are represented with continuous arrows. However, MMM also detects that the students deviated from the prescribed Map, since some

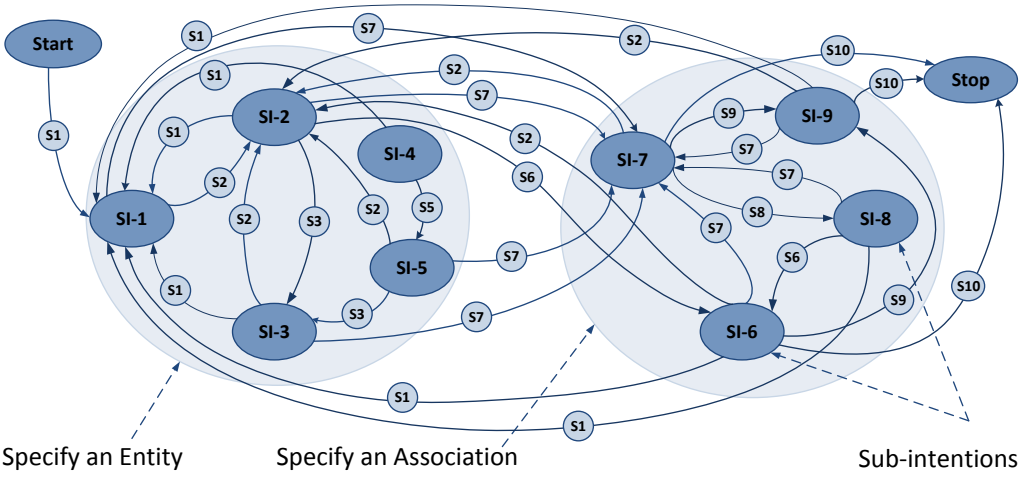

Figure 4.18: Map Process Model discovered by Deep Miner algorithm and supervised learning ($\varepsilon = 0.05$)

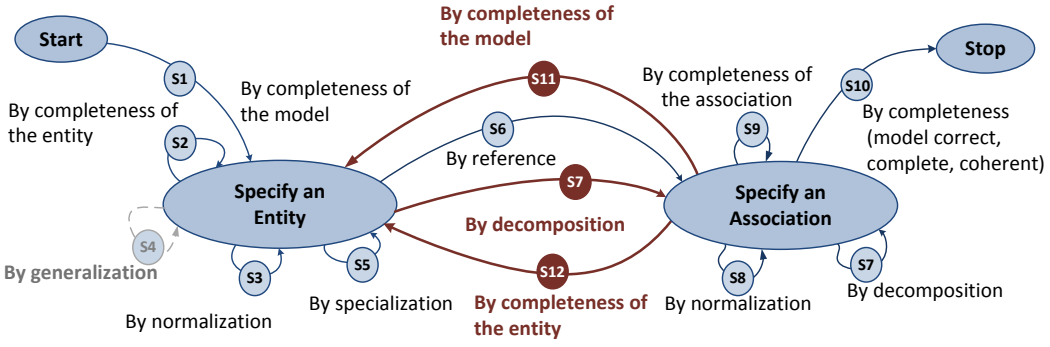

Figure 4.19: High-level of granularity for Map Process model discovered by Map Miner algorithm and supervised learning.

strategies are not used as they were prescribed. By analyzing and comparing the discovered Map with the prescribed Map, the observations are the following :

- The first ascertainment concerns the matches between the prescribed and discovered Maps. The students chose strategy S_1 to achieve the intention **Specify an entity** and they continued to try to reach this intention by choosing strategies S_2, S_3 and S_5. To fulfill the intention **Specify an association**, they chose strategy S_7, S_8 and S_9. Finally, they chose strategy S_{10} to **Stop** the process.

- The second ascertainment shows there are one mismatch between the prescribed and discovered Maps. The students never enacted S_4: by generalization and S_7 is chosen in the wrong section. Indeed, the strategy S_7 must be selected only in the section <Specify an association, Specify an association, By decomposition>. However, this strategy is also selected between the intentions **Specify an entity** and **Specify an association**.

- Some strategies shown in Figure 4.18 raise some issues. The section <Specify an association, Specify an entity, By completeness of the association> is not coherent as the target intention completely differs from what is implied by the strategy. This section does not represented in Figure 4.19. The section <Specify an association, Specify an entity, By completeness of the model> was kept in Figure 4.19 as users can complete the model by creating a new entity after specifying an association. In this case, the situation takes into account the modified product (the E/R diagram), on the contrary of the section <Start, Specify an entity, By completeness of the model> which implies no existing product to take into account. However, it is not possible to tell from the traces if the users properly followed these strategies according to the product situation (existing E/R diagram or not). With the same reasoning, the section <Specify an association, Specify an entity, By completeness of the entity> is taken in Figure 4.19 allowing users to add attributes to an entity of their diagram after specifying an association.

4.5.2 MMM Using Unsupervised Learning

4.5.2.1 Estimating Model Parameters

By applying unsupervised learning on the same activities traces a different Map process model is discovered. We recall that in this case, no labeling process for the strategies is necessary to run the learning algorithm. Consequently, only the traces of activities are used as inputs of BWA. However, as mentioned earlier, the BWA must be initialized with a number of possible strategies. Here, the right number of strategies is found by the proposed heuristic method detailed in Section 4.3.2.4. Figure 4.20 shows that above ten possible strategies, BWA always gives the same ten strategies. This is because the BWA produces an emission matrix with several identical strategies when the number of possible strategies is too high. Hence, 10

is the number of strategies for the model with the highest probability to generate the observed traces. Once that the number of strategies is fixed, the BWA can be executed to learn the model parameters.

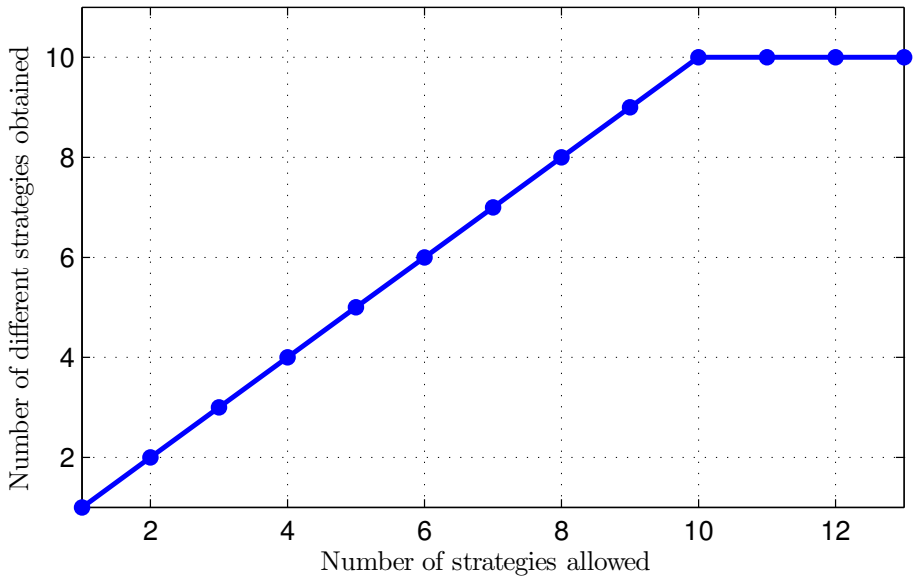

Figure 4.20: Number of strategies discovered by the heuristic method

4.5.2.2 Applying Deep Miner Algorithm

Deep Miner algorithm generates the pseudo-Map by maximizing the metric fitness and precision (4.17). Indeed, the transition matrix allows knowing the position of each strategy on the Map, *i.e.* what is the source intention and the target intention of each strategy. The Map obtained by Deep Miner algorithm with $\varepsilon = 0.05$ is depicted on Figure 4.21.

4.5.2.3 Applying Map Miner Algorithm

To obtain a Map process model, Map Miner algorithm groups the sub-intentions into 4 high-level intentions as shown on Figure 4.22 and the discovered strategies are detailed in the right side of Table 4.5.2.3.

On the contrary to supervised learning, since no prior information about strategies is available, the names of strategies and intentions are not known. However, from the emission matrix **E**, each strategy discovered by MMM can be associated to some activities. Based on the names of activities, it is then possible to infer the main topics of the strategies through a semantic analysis (see Table 4.8). In the same way, the intentions names can be inferred by analyzing the strategies leading to each intention.

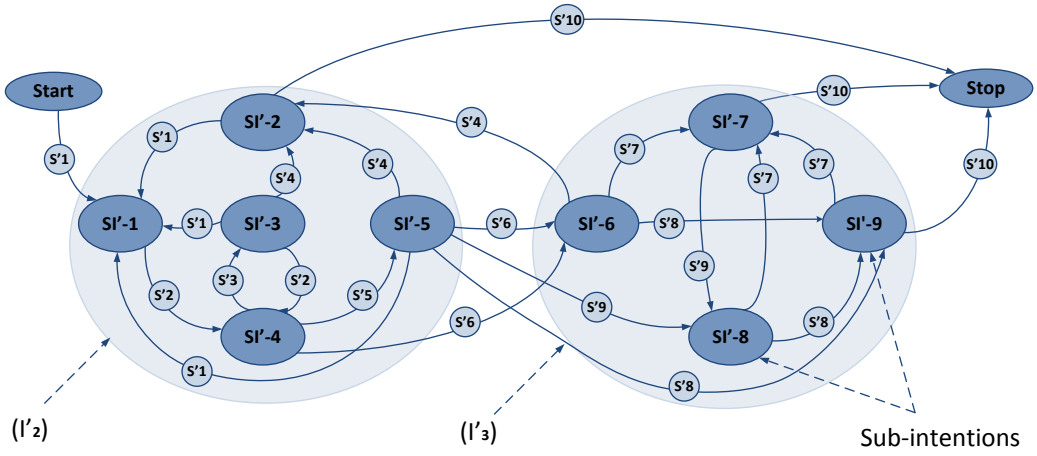

Figure 4.21: Map Process Model discovered by Deep Miner algorithm and unsupervised learning ($\varepsilon = 0.05$)

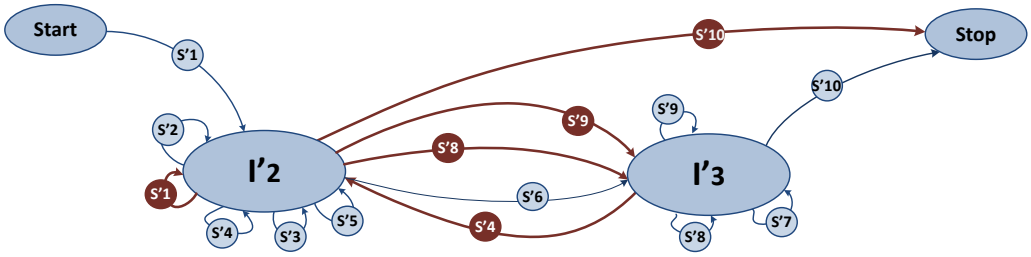

Figure 4.22: High-level of granularity for Map process model discovered by Map Miner algorithm and unsupervised learning

Figure 4.23 depicts a comparison of the likelihood (4.11) of the prescribed Map, the Map discovered by maximizing (4.17), and the unsupervised parameters (*i.e.*, the parameters directly obtained from of the emission and transition matrices using BWA). The likelihood of a Map represents the probability that the Map generates the observed traces. The likelihood of unsupervised parameters represents the probability that the parameters estimated by unsupervised learning generates the observed traces. A first observation is that the likelihood of the unsupervised parameters is higher than the prescribed Map or the discovered Map. This means the probability that the estimated parameters generate the observed traces is higher than the prescribed Map. This phenomenon is normal since, by definition, the result of BWA maximizes the likelihood. A second observation is that the likelihood of the discovered Map is lower than the matrices generated by BWA. This is due

Strategies Index	Strategies Topics discovered by Matrix E
S'_1	entity, creation, specify
S'_1	entity, creation, specify
S'_2	attribute, entity, creation
S'_3	entity, delete, creation, specialize
S'_4	delete, creation, attribute, entity, association
S'_5	primary key, creation, entity
S'_6	creation, entity, attribute, link
S'_8	association, entity, link, attribute, creation
S'_9	creation, association, entity, attributes
S'_4	delete, creation, attribute, entity, association
S'_7	link, creation, delete, entity, association
S'_8	association, entity, link, attribute, creation
S_9	creation, association, entity, attributes
S'_{10}	check, model, coherent
S'_{10}	check, model, coherent

Table 4.8: Strategies topic discovered by matrix **E** for unsupervised learning of the example

to the constraints of the structure imposed by the topology of a Map (*e.g.*, the transition from **Stop** to the **Start** is not allowed). However, the discovered Map has a higher likelihood than the prescribed Map, which is a convincing result since it means that in terms of fitness of the observed activities, the discovered Map is a better Map than the prescribed Map.

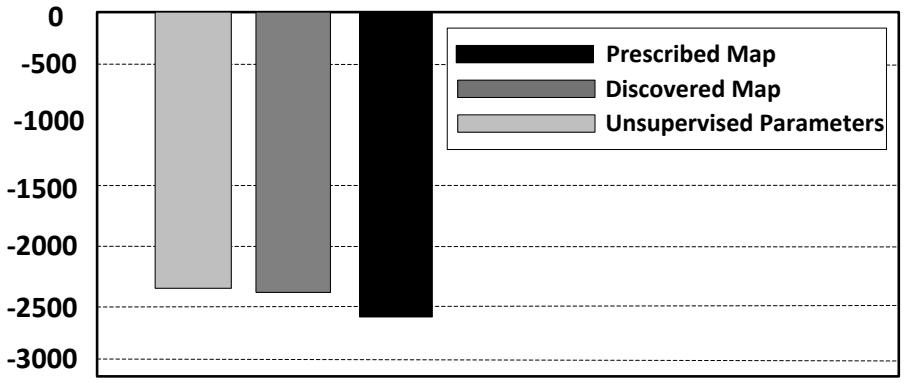

Figure 4.23: Comparison of the likelihood of the prescribed Map, the discovered Map, and the unsupervised parameters

Finally, Figure 4.24 shows the effect of the choice of ε on the likelihood and the number of sections of the discovered Map. When ε is close to 0, almost all the transitions from the unsupervised model are present in the discovered Map. Consequently, the likelihood of the discovered Map is high but the Map is hardly understandable by humans since it has too many sections. However, when ε increases, the number of sections, as well as the likelihood of the discovered Map, decrease. The Map gets more easily understandable by humans but it is not as accurate in terms of transitions. This study demonstrates that the value of ε has to be set to obtain a trade-off between granularity of the Map and its understanding.

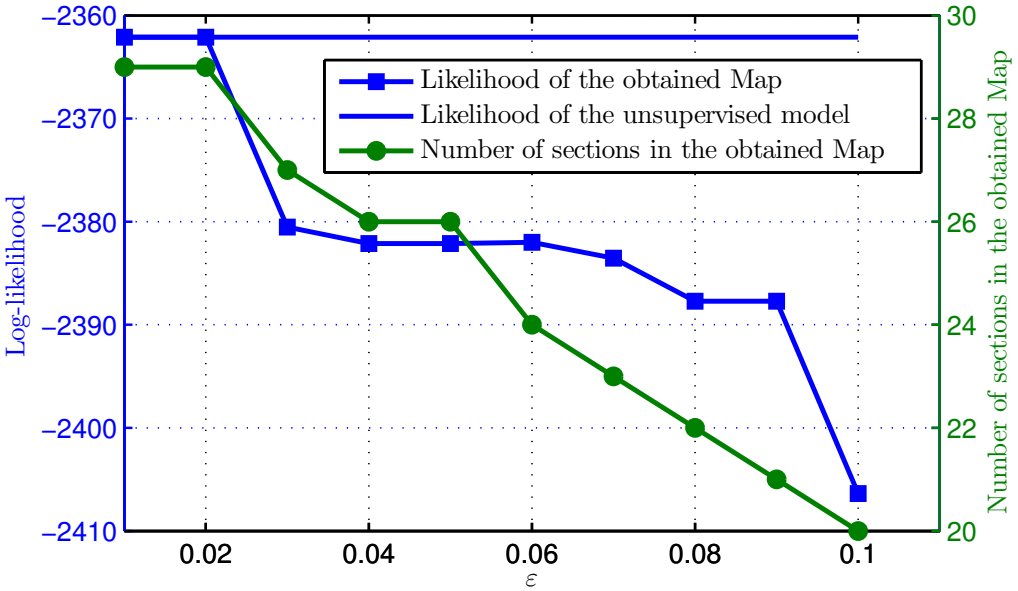

Figure 4.24: Likelihood and number of sections of the discovered Map with regards to ε for the E/R traces

By analyzing and comparing the prescribed Map and the pseudo-Map discovered by unsupervised learning, some observations are:

- There is a number of matches between the new groups of strategies and the groups of strategies in the prescribed Map, such as S_1 and S'_1, which both start the process and are made of activity a_1. The Strategy S_{10} and S'_{10} are also similar in both Maps. They end the process and are made of a_{12}. Another match is between S_2 and S'_2, which are made of activity a_2. This was expected since the students had to follow the prescribed Map.
- The few mismatches between the prescribed and discovered strategies indicate that some prescribed strategies were not followed as intended by the students. The discovered strategies permit learning how the students preferred to behave. For instance, in the prescribed Map, activity a_2 is only

present in strategy S_2 for intention I_2 whereas in the discovered Map, it is present in many strategies: S'_1, S'_4, S'_6, S'_7, and S'_8 for intentions I'_2, and I'_3. The same phenomenon is true for activity a_1. It means that these particular activities are used by students during the entire process and not only for a single intention. This knowledge can be used to (i) improve the software used by the students to make the prescribed model easier to follow, (ii) modify the prescribed model by taking into account to the process the students actually followed.

- Most activities are related to the same intentions in the prescribed and the discovered Maps. This is true for activities a_3, a_4, a_6, and a_7 which are only related to I_2 and I'_2. This is also true for activities a_9 and a_{11} which are related to I_3 and I'_3. However, a_{10} is very seldom used by students and does not appear in the discovered strategies.

Note that $I_i \rightarrow I_j$ expresses the transition from the intention I_i to the intention I_j.

Prescribed Map Process Model				Discovered Map process Model		
Intentions	**Index**	**Name of Strategies**	**Activities**	**Intentions**	**Index**	**Activities**
$I_1 \rightarrow I_2$	S_1	By completeness (model)	a_1	$I'_1 \rightarrow I'_2$	S'_1	a_1 (0.94)
				I'_2	S'_1	a_1 (0.94)
I_2	S_2	By completeness (entity)	a_2		S'_2	a_2 (0.88), a_1(0.09)
	S_3	By specialization	a_6, a_5, a_7		S'_3	a_5 (0.1), a_6 (0.63), a_7 (0.28)
	S_4	By generalization	a_1, a_3		S'_4	a_1 (0.11), a_2 (0.54), a_8 (0.25)
	S_5	By normalization	a_1, a_4		S'_5	a_2 (0.09), a_3 (0.13), a_4 (0.39), a_5 (0.40)
$I_2 \rightarrow I_3$	S_6	By reference	a_8, a_9	$I'_2 \rightarrow I'_3$	S'_6	a_1 (0.15), a_2 (0.79)
					S'_8	a_1 (0.09), a_2 (0.81), a_9 (0.08)
					S'_9	a_1 (0.37), a_9 (0.19), a_{11} (0.34)
				$I'_3 \rightarrow I'_2$	S'_4	a_1 (0.11), a_2 (0.54), a_8 (0.25)
I_3	S_7	By decomposition	a_9	I'_3	S'_7	a_9 (0.83), a_1 (0.05), a_2 (0.05)
	S_8	By normalization	a_{10}		S'_8	a_1 (0.09), a_2 (0.81), a_9 (0.08)
	S_9	By completeness (asso.)	a_{11}		S'_9	a_1 (0.37), a_9 (0.19), a_{11}(0.34)
$I_3 \rightarrow I_4$	S_{10}	By completeness (final)	a_{12}	$I'_3 \rightarrow I'_4$	S'_{10}	a_{12} (0.87), a_9 (0.08)
				$I'_2 \rightarrow I'_4$	S'_{10}	a_{12} (0.87), a_9 (0.08)

Table 4.9: Strategies of the prescribed Map (left) and the discovered Map (right) for the example

4.5.3 Discussion and Threats to Validity

It is interesting to discuss the results discovered in the previous section from both quantitative (models likelihood, algorithm convergence, complexity) and qualitative (models interpretation) points of view.

- Adopting a qualitative point of view, although some strategies from the prescribed Map and the Map discovered by unsupervised learning are similar (S_1 and S'_1, S_2 and S'_2, S_{10} and S'_{10}, S_7 and S'_7), most strategies from unsupervised learning cannot be exactly identified to prescribed strategies. It is not due to a poor compliance of the Map discovered by unsupervised learning but due to the supervised learning assumption, *i.e.*, the prescribed Map is actually followed by students. This assumption is not true. Indeed, during the enactment of the process, students may deliberately or accidentally not follow the prescribed Map. Consequently, assuming that the prescribed model is followed by students, this creates a bias in the definition of strategies and intentions. In addition, there is no ground truth for labeling the activities sequences. Consequently, the labeling could be flawed as it is a subjective process. Moreover, assigning the labels to the strategies and intentions constrains the discovered Map to a limited space which leads to poor performance of supervised learning. This phenomenon can be verified with the deviations of students detected by discovered Maps. Whereas the Map discovered by supervised learning detected only two deviations (S_4 and S_7), the Map discovered by unsupervised learning detected five deviations which are not the same as the supervised learning ones (S'_1, S'_4, S'_8, S'_9, S'_{10}).

- The log-likelihood of the parameters estimated by unsupervised learning in the experiment is higher than supervised learning. This phenomenon remains true when Maps are extracted from the estimated parameters: Figures 4.25 and 4.26 provide the log-likelihood of Maps discovered with unsupervised learning with regards to the probability threshold ε, while the log-likelihood of Maps discovered with supervised learning is equal to $-\infty$. In other words, Maps discovered by supervised learning cannot generate all the traces of activities in both trace under study, while Maps discovered by unsupervised learning do. Interestingly, the number of sections of the Maps discovered by unsupervised learning is also lower than the Maps discovered by supervised leaning, which makes the Maps structure simple and easier to understand by human.

- From a cost-benefit and human-centric point of view, cognitive tasks are time-consuming and labor intensive. Thus, the cost of labeling the data for supervised learning approach is quite high as it involves the students' commitment to label and comment their activities at each step of the process. In comparison, the only human effort for unsupervised learning is to choose the number

of strategies for the intentional process model. Nevertheless, the unsupervised learning requires a minimal human intervention to learn the parameters and it allows obtaining intentional process models that match the actual enacted process. The drawbacks of unsupervised learning are a higher computation complexity and the need to automate the naming of discovered strategies and intentions.

- The BWA cannot guarantee to converge to the global maximum likelihood (see Section 4.3.2.2). The convergence depends on the initialization of the matrices **T** and **E** and it converges at 9,986 learning iterations for the example. The supervised algorithm converges in the first iteration. These iterations make unsupervised learning a more expensive method than supervised learning.
- While the complexity of the BWA is high due to its requirement to several iterations until the convergence to a local optimum, complexity of supervised learning is very low.

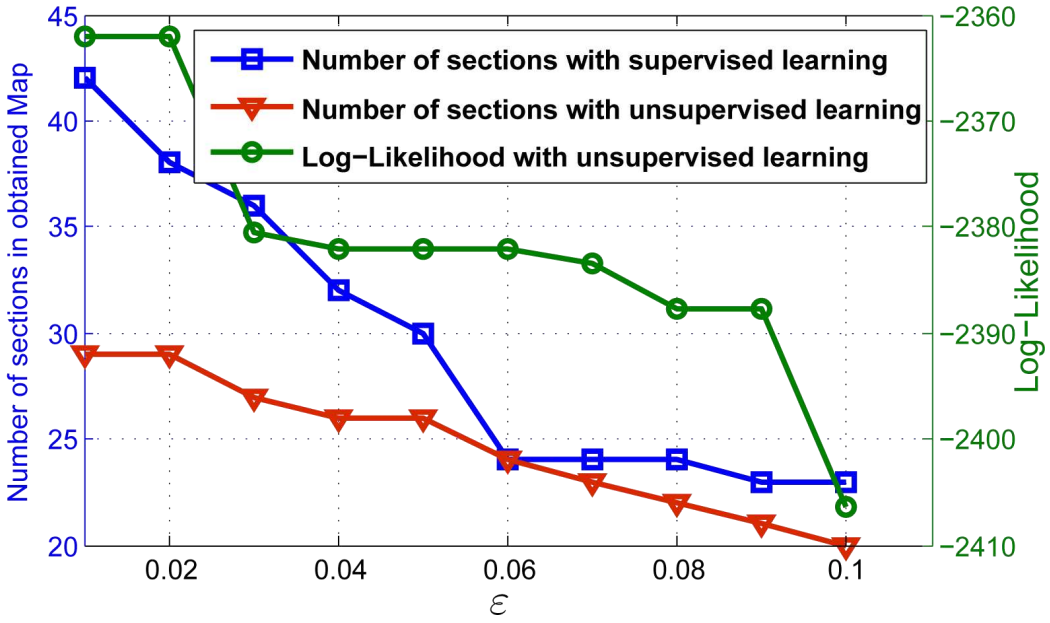

Figure 4.25: Log-likelihood and number of sections of the discovered Maps with regards to ε for the example

Table 4.10 presents a practical comparison of both learning approaches for the example. The supervised learning takes as input 66 traces of activities and 66 related labeled strategies. The unsupervised learning takes as input only 66 traces of activities. Whereas the convergence speed of supervised learning is very fast, since it requires only 1 iteration, the convergence speed of unsupervised learning is slow, since it requires 9,986 iterations. The likelihood of supervised learning is $-2.54e^3$

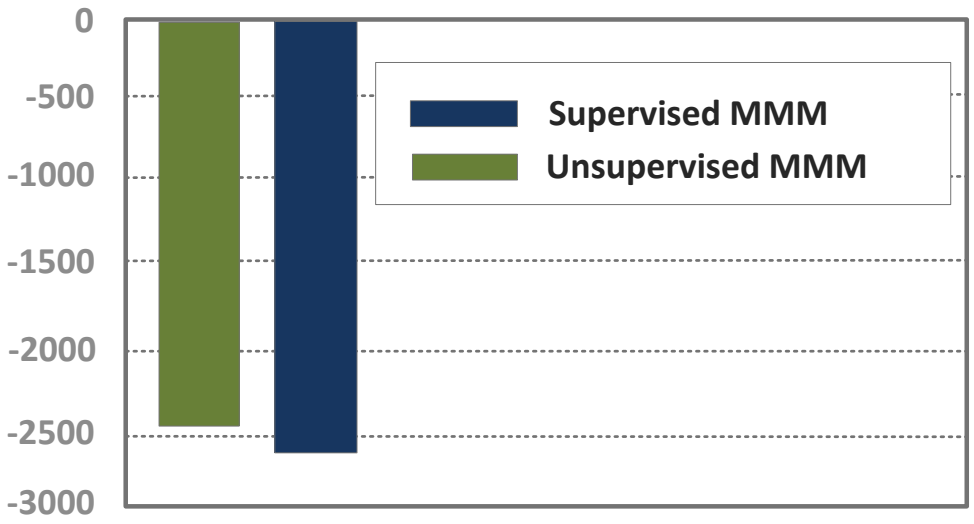

Figure 4.26: Log-likelihood of supervised and unsupervised

(logarithmic scale), which is lower than the likelihood of unsupervised learning ($-2.36e^3$), which demonstrates the results provided by unsupervised learning reflect most likely the actual process.

Learning Approach	**Traces for learning**	**A-priori knowledge**	**Convergence speed (complexity)**	**The estimated parameters Log-likelihood**
Supervised learning	66 traces of activities, 66 traces of strategies	set of activities, set of strategies	1 iteration	$-2.54e^3$
Unsupervised learning	66 traces of activities	set of activities, number of strategies	$9,986$ iterations	$-2.36e^3$

Table 4.10: Practical comparison of supervised and unsupervised learning on the traces of the example

4.6 Validating the Method for the Discovery of Map Path

In order to evaluate the results that are provided by application of the VA on the example data, the *Recall*, *Precision* and *F-score* (*i.e.*, a combination of recall and precision) measures [Goutte 2005] are chosen. Before explaining these measures, a brief overview of some terms is necessary:

- The True Positive (TP) represents the number of strategies correctly assigned by the VA as belonging to the right class of strategy.
- The False Negative (FN) represents the number of strategies, which were not assigned to the right class of strategy.
- The False Positive (FP) represents the number of strategies incorrectly assigned to the class of strategy. The accuracy of the VA prediction can be evaluated by checking if the prediction matches to the actual strategies.
- The True Negative (TN) represents the number of strategies correctly assigned to the right class of strategy.

Table 4.11 shows these definitions.

		Strategies	
		True	**False**
Prediction	**True**	True Positive (TP) (Correct result)	False Positive (FP) (Unexpected result)
	False	False Negative (FN) (Missing result)	True Negative (TN) (Absence of correct result)

Table 4.11: Prediction of the strategies by VA

Recall is the ratio between the number of strategies correctly identified by the VA and the number of strategies in the trace. Note that it does not take into account the number of strategies falsely identified by the algorithm. Recall is defined by the following expression:

$$Recall = \frac{TP}{TP + FN} \tag{4.23}$$

Precision denotes the ratio between the number of strategies correctly identified by the VA and the number of strategies identified by the algorithm. Precision is defined by:

$$Precision = \frac{TP}{TP + FP} \tag{4.24}$$

In general, it is possible to increase recall to reduce precision and vice versa. F-score is a combination of precision and recall:

$$Fscore = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.25}$$

This is also known as F_1 measure, because precision and recall are weighted equally. F_1 measures the effectiveness of strategy recovery, considering the same importance for recall and precision.

The recall, precision and F-score of the Supervised Map Miner method is calculated for all strategies. Six strategies S_1, S_2, S_6, S_7, S_9, and S_{10} had a score almost near 100%, which mean they could systematically be correctly retrieved from observations of students' activities. These strategies are shown in Figures 4.28, 4.30, and 4.32.

The first observation is that recall and precision are stabilized when the estimation sequence length reaches its maximum value. It means that for a length of 66, the VA provides stable results. Table 4.12 shows recall, precision and F-score for all the mined strategies from the 66 traces. For example, the algorithm finds 99% of activities related to strategy S_1. This means that almost all the activities associated to strategy S_1 were identified. Now the question is: does the algorithm associate several activities to the strategies while, in fact, they belong to other strategies? This question could be addressed using the precision ratio. For example, the precision result stabilizes at 99% for strategy S_1, which means only 1% of the activities are associated to strategy S_1 while they should not.

Strategy	Recall	Precision	F-score
1	99%	99%	0.99
2	100%	100%	1
3	95%	92%	0.94
4	72%	65%	0.68
5	87%	92%	0.90
6	96%	94%	0.95
7	98%	100%	0.99
8	73%	78%	0.75
9	96%	100%	0.98
10	100%	100%	1

Table 4.12: Recall, Precision, F-score for all the mined strategies

Figures 4.27, 4.29, and 4.31 depict respectively, recall, precision and F-score in terms of number of training traces for strategies S_1, S_3, S_4, S_5, and S_8. The same measures were taken for all the strategies and reported in Figure 4.33, which shows the global performance of traces estimation. In this experiment, the curve of recall starts from 0.9114 and its value increases until reaching the maximum value at 0.9240. The curves of precision and F-score start from 0.9120 and 0.8869 and stop at 0.9194 and 0.9207, respectively. These results demonstrate that the accuracy of retrieval for the ten strategies is 0.9207, which indicates that the algorithm has found the right strategies corresponding to the traces of activities with a reliability of 92%.

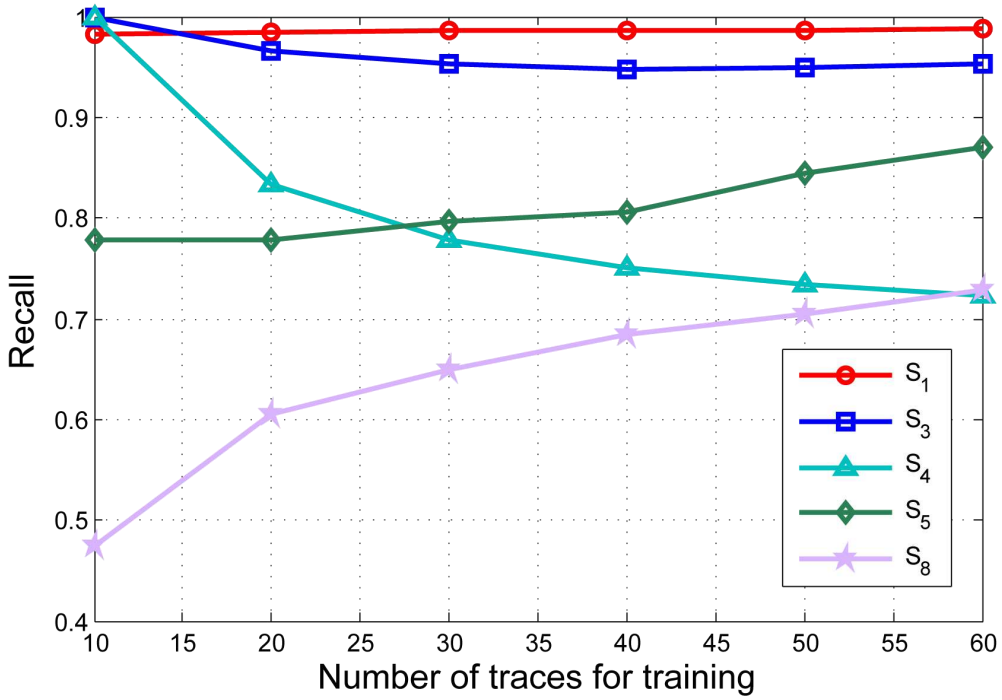

Figure 4.27: Recall for Strategies S_1, S_3, S_4, S_5, and S_8.

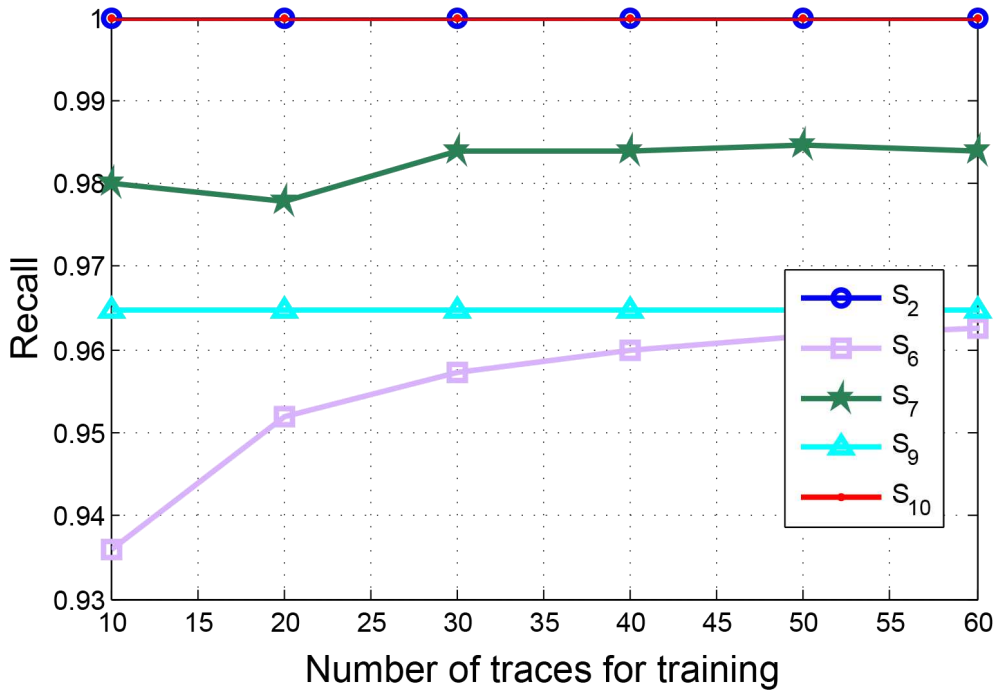

Figure 4.28: Recall for Strategies S_2, S_6, S_7, S_9, and S_{10}.

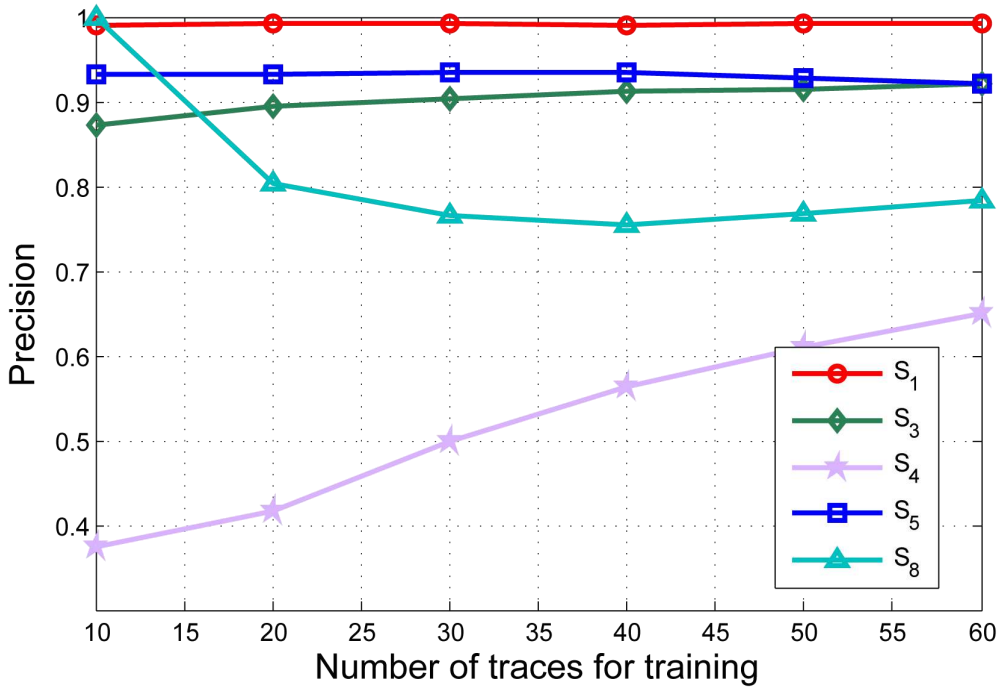

Figure 4.29: Precision for Strategies S_1, S_3, S_4, S_5 and S_8.

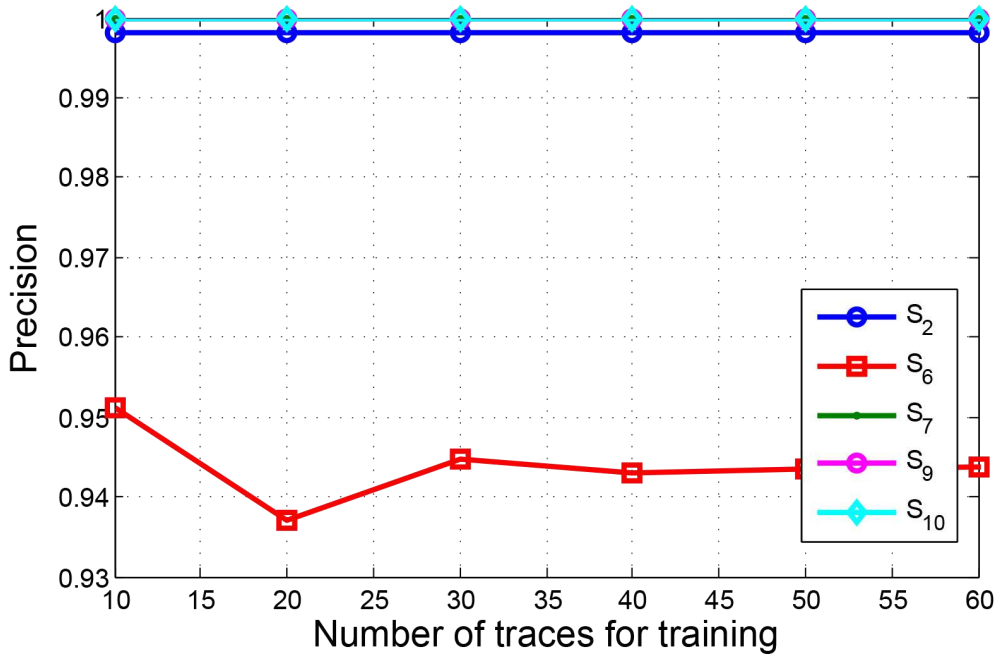

Figure 4.30: Precision for Strategies S_2, S_6, S_7, S_9, and S_{10}.

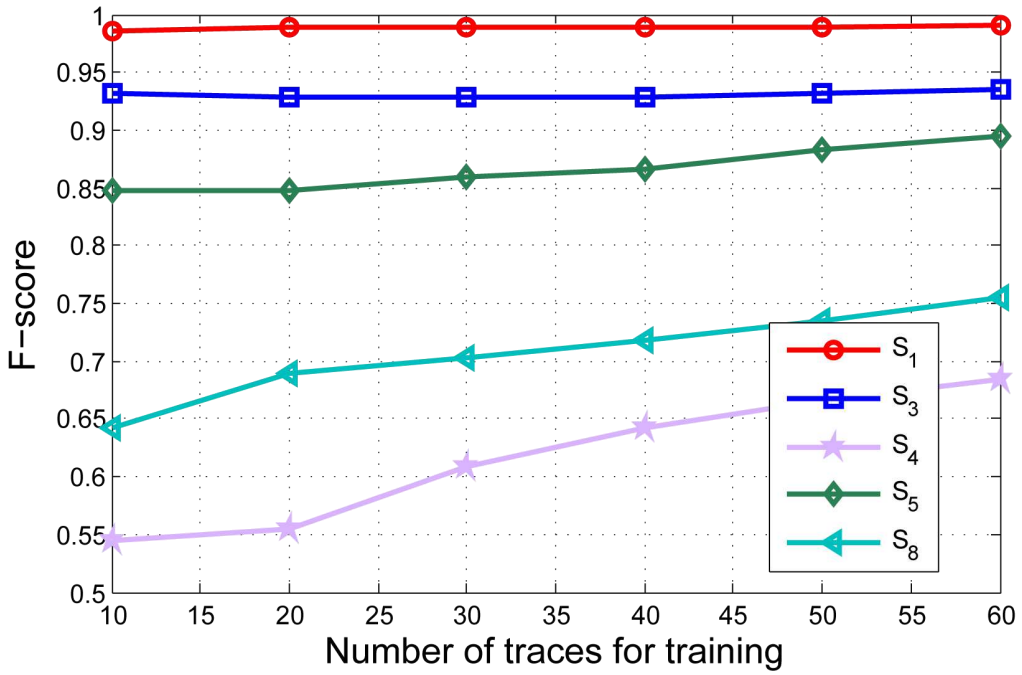

Figure 4.31: F-score for Strategies S_1, S_3, S_4, S_5 and S_8.

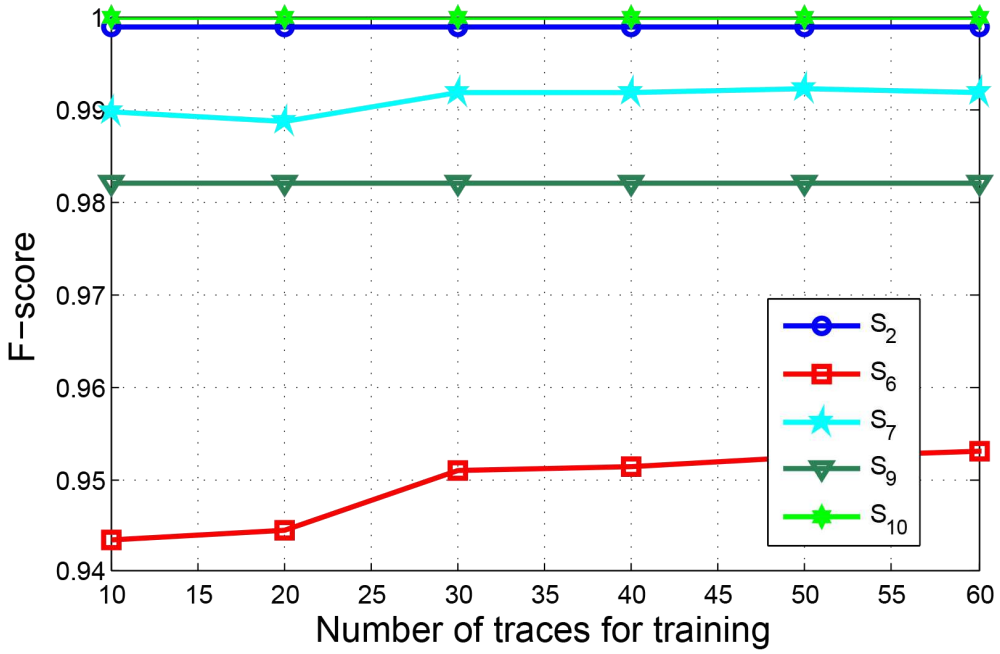

Figure 4.32: F-score for Strategies S_2, S_6, S_7, S_9, and S_{10}.

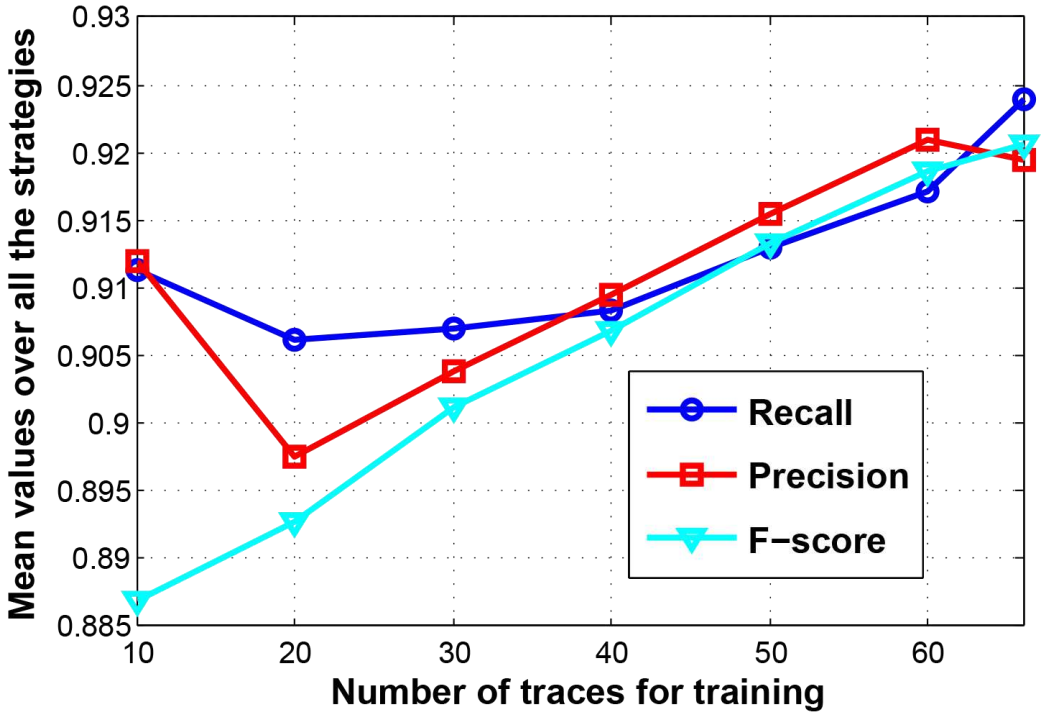

Figure 4.33: Mean value over all the strategies

4.7 Conclusion

This chapter presented in detail MMM, which allows semi-automating the construction of Map process models from users' traces. MMM consists of three main stages: (i) first, it estimates the users' strategies from users' traces, (ii) second, it constructs a pseudo-Map from the estimated strategies, (iii) third, it clusters sub-intentions in the pseudo-Map into a Map. The parameters of an HMM, *i.e.*, the estimated strategies (**T**) and the occurrence of each activity in each estimated strategies (**E**), can be estimated either by supervised learning or unsupervised learning. The theoretical and practical synthesis of both learning approaches demonstrates their pros and cons. On the one hand, several issues hinder the application of supervised learning in modeling humans' cognitive process, such as considerable humans' involvement in terms of data labeling, introducing inherent humans' biases and lack of accurate ground truth. Moreover, the likelihood of estimated strategies obtains by supervised learning is lower than the likelihood of unsupervised learning. On the other hand, unsupervised learning offers a higher likelihood, which means the estimated parameters match better the reality (what really happened in the trace). Furthermore, unsupervised learning requires lower humans' involvement since it does not need the labeling procedure. The application of unsupervised learning on an example validates all the aforementioned points. Therefore, to automate the construction of Map process models, unsupervised learning seems a more promising approach.

Both Deep Miner and Map Miner algorithms are applied on the example with students. The likelihood of the produced Map process models are evaluated. The likelihood (4.11) of the different HMM models studied in this example to generate the traces of the students: the prescribed Map, the Map obtained by maximizing (4.17), and the unsupervised Map, directly made of the emission and transition matrices using BWA. A first observation is that the likelihood of the unsupervised model (*i.e.*, the model parameters obtained by unsupervised learning) is higher than the prescribed Map or the obtained Map. This is normal since, by definition, the result of BWA maximizes the likelihood. A second observation is that the likelihood of the obtained Map is lower than the matrices generated by BWA. This is due to the constraints of structure imposed by the topology of a Map. However, the obtained Map has a higher likelihood than the prescribed Map, which is a satisfying result since it means that in terms of fitness of the observed activities, the obtained Map is a better Map than the prescribed Map.

As mentioned in Section 4.4 once the parameters of HMMs (matrices **E** and **T**) are estimated, it is possible to find the most likely strategies related to a given sequence of activities by VA. This allows identifying the path taken by users. The results of the application of VA on the example show the efficiency of the algorithm. It can find the right strategies corresponding to the traces of activities with a high reliability.

The proposed method will be applied on a case study in next Chapter 5.

Chapter 5

Validation of the Proposed Method

Contents

5.1 Case Study: Usage Data Collector of Eclipse

5.1.1 Presentation of the Case Study

This case study demonstrates the capability of MMM to handle the large-scale data along with providing comprehensive and reliable results.

Eclipse Usage Data Collector (UDC) [Eclipse 2013] is a system which collects information about how developers use the Eclipse platform. The Eclipse Foundation provides these data to help committers and organizations to better understand how the community makes use of Eclipse [UDC 2013]. In this perspective, this case study aims at modeling the UDC developers' activities in terms of intentions and strategies by constructing automatically a Map process model that is actually followed by developers. The discovered Map process model is useful to analyze developers' behaviors (see 5.1.3).

5.1.1.1 Usage Data Collector Event logs

The event logs of this case study are event logs of developers who committed their activities to Usage Data Collector (UDC) of Eclipse [Eclipse 2013]. The event logs

are uploaded to servers hosted by the *Eclipse Foundation*. The event logs contains 1,048,576 event logs from developers who agreed to send their data back to the Eclipse Foundation. These data aggregate activities from over 10,000 Java developers between September 2009 and January 2010. The activities are recorded by timestamps for each developer, which allows knowing when and by whom, activities were committed.

5.1.1.2 Developers' Activities

In order to apply MMM, it is important to prepare the event logs. The number of unique developers' activities per month exceeds 500 activities. This number contains both the recurring activities and the non-recurring activities (*i.e.*, activities which are not frequently performed by developers). The non-recurring activities activities are not representative of the developers' behavior characteristics because they have not been repeated enough to be a behavioral-pattern. For this reason, and also for readability, the case study is limited to the 130 most frequent activities performed by developers. Table 5.1 contains the list of these activities. Some of these activities are the commands performed directly by developers; some of them are the frameworks, plug-ins or built-in features of Eclipse used by developers during their development process. For readability reasons, the prefix *org.eclipse* of the activities is removed. The plug-ins and frameworks are shown in bold letters and the related activities are inside brackets. For each developer who submitted his/her trace of activities to UDC, the trace containing only activities among the 130 most frequent is extracted. In this way, the event logs is prepared to apply MMM.

5.1.2 Applying MMM on the Traces

Once the traces is ready, BWA estimates the transition matrix (developers' strategies). Note that, once again the number of strategies obtained by the heuristic method for this case study is 10. The strategies are represented in Table 5.1 with their corresponding groups of activities. Figure 5.1 depicts the effect of the choice of the threshold ε on the likelihood and the number of sections of the obtained Map. As mentioned earlier, ε expresses the level of granularity for a Map. An expert can choose the value of ε regarding the expected level of granularity. In this case study, the value of ε is set to 0.06 to have a good trade-off between having a likelihood with a relative high value and a reasonable number of sections. Finally, the Map obtained by Deep Miner and Map Miner algorithms is shown on Figure 5.2 and 5.3, respectively.

Regarding the obtained Map, 22 sub-intentions are grouped by Map Miner algorithm into 7 groups of high-level intentions. Note that MMM can discover accurately the beginning and the end of a process; thus the intentions **Start** and **Stop** are clearly determined on the obtained Map. The transition probabilities from one intention to a strategy are annotated on the arrows. These values correspond to

the probabilities that the developers selected a strategy from a given intention. The values on the loops indicate, the probabilities that the developers continued to perform the activities related to the looped strategies.

5.1.2.1 Strategies and Intentions Naming Procedure

MMM discovers the strategies and where that lead to intentions. In other words, MMM reconstructs the topology of the Map process model from traces; the names of the strategies and intentions are not fully automatically generated. Nevertheless, it is possible to infer the names of strategies and intentions from the emission matrix **E**. Indeed, **E** specifies the activities associated to each strategy discovered by the MMM (see Table 5.1). Therefore, based on the names of the activities grouped into a strategy, it is then possible to manually infer the names of the strategies through a semantic analysis of their properties and interrelationships. In the same way, since the strategies lead to intentions, the names of intentions can be inferred by analyzing the strategies leading to each intention. For instance, the main activities grouped into the strategy S_5 are 'refactoring.commands', 'jdt.junit', 'debug.ui.commands', etc. From these activities one can infer the developers wanted to debug, to refactor and to test the code; thus, the name inferred for this strategy is *by refactoring, testing and debugging*. Further, the main activities for strategy S_6 are 'delete', 'paste', 'copy', 'undo', etc. This means the developers wanted to modify a code or a file; the name inferred for this strategy is *by file modification*. Since both strategies lead to an intention, it is possible to infer that the developers who performed S_5 and S_6 intended to **Fix a bug**. By applying this procedure, the names of all strategies (denoted on the arrows) and intentions are inferred. Table 5.2 represents the topics obtained by **E** and the inferred strategies names. The inferred names of intentions are **Start, Initiate the development, Manage tasks, Fix a bug, Improve the code, Commit the code**. This naming protocol remains to be fully automated by building sophisticated ontologies.

5.1.3 Analysis of Eclipse Developers' Behavior

Discovering the Map for developers of Eclipse UDC allows understanding the developers' behaviors during the development process. As shown in Figure 5.3, they have selected different paths (sequences of strategies) with different probabilities to fulfill their intentions. An expert can analyze these behaviors in order to understand how, why and with which probabilities developers make use of different components or plug-ins of Eclipse: where they follow the *best practice* of software development projects and where they deviate from these rules, which components or plug-ins are more involved than the others, which paths are more/less taken or where are the system problems, etc. The Map can also be used to provide recommendations to developers in order to choose the best path to fulfill his/her intentions. In this case study, there is no prior model to compare with the obtained Map. Hereafter, some observations are detailed to address some of the aforementioned claims in practice.

Table 5.1: Strategies index and related activities for UDC Eclipse

Strategies Index	**Activities Names**
S_1	**mylyn.tasks.ui.commands.**[OpenTask, AddTaskRepository, ActivateTask, SearchForTask], **mylyn.context.ui.commands.**[Open.context.dialog, AttachContext, interest.Increment, interest.Decrement], **mylyn.monitor.ui**, **mylyn.team.ui**
S_2	**core.**[jobs, net, filesystem, resource, runtime, variables, contenttype, databinding.observable], **equinox.p2.ui.sdk.install**
S_3	**mylyn.context.ui.commands.**[Open.context.dialog, AttachContext, interest.Increment, interest.Decrement],**team.cvs.ui.**[branch, replace, GenerateDiff, ShowHistory, Add, Tag,merg, compareWithTag], **jsch.core**, **mylyn.**[monitor.ui, team.ui, commons.ui, bugzilla.ui]
S_4	**pde.ui.**EquinoxLaunchShortcut.run, **equinox.p2.ui.**sdk.update, **equinox.**[ds, simpleconfigurator.manipulator, frameworkadmin, app, common, directorywatcher, engine, core, metadata.repository, garbagecollector, ui.sdk.scheduler, repository, preferences, exemplarysetup, registry, updatechecker]
S_5	**core.**[databinding.observable, core.net, core.filesystem, core.resource, core.runtime, core.variables, core.contenttype], **debug.ui.commands.**[RunLast, Debuglast, eof, StepOver, TerminateAndRelaunch, execute, AddBreakPoint, TogglebreakPoint],**jdt.debug.ui.** [commands.Execute, commands.Inspect], **jdt.junit.**[junitShortcut.rerunLast, gotoTest,junitShortcut.debug], **ltk.ui.refactoring.commands.** [deleteResources, renameresources, moveResources], compare.selectPreviousChange
S_6	**ui.edit.**[delete, paste, copy undo, text.goto.lineEnd, text.contentAssist.proposals, text.goto.wordNext], **ui.file.save**
S_7	cdt.ui.editor, **jdt.junit.**[junitShortcut.rerunLast, gotoTest, junitShortcut.debug], **team.cvs.ui.**[CompareWithRevision, CompareWithLatestRevisionCommand, CompareWithWorkingCopyCommand],**ui.edit.**[delete, paste, copy undo, text.goto.lineEnd, text.contentAssist.proposals, text.goto.wordNext]
S_8	**team.ui.**[synchronizeLast, TeamSynchronizingPerspective, synchronizeAll, applyPatch], **ltk.core.refactoring.refactor.**[create.refactoring.script, show.refactoring.history]
S_9	**mylyn.monitor.ui**, **mylyn.context.ui**, **mylyn.commons.ui**, **team.cvs.ui.**[commitAll, Commit, CompareWithRemote, Sync]
S_{10}	**mylyn.monitor.ui**, **mylyn.bugzilla.core**, **mylyn.bugzilla.ui**, **team.cvs.ui.**[commitAll, Commit, CompareWithRemote, Sync]

Table 5.2: Strategies index, topics and inferred strategies names for UDC Eclipse

Strategies labels	**Topics obtained by E**	**Inferred strategies names**
S_1	OpenTask, AddTaskRepository, ActivateTask, SearchForTask, Open.context.dialog, AttachContext, interest.Increment, interest.Decrement, mylyn.monitor.ui, mylyn.team.ui	By project tracking and team planning
S_2	jobs, net, filesystem, resource, runtime, variables, contenttype, databinding.observable, equinox.p2.ui.sdk.install	By regular programming activities
S_3	Open.context.dialog, AttachContext, interest.Increment, interest.Decrement, branch, replace, GenerateDiff, ShowHistory, Add, Tag,merg, compareWithTag, jsch.core, monitor.ui, team.ui, commons.ui, bugzilla.ui	By code/task sharing
S_4	EquinoxLaunchShortcut.run, equinox.p2.ui.sdk.update, simpleconfigurator.manipulator, frameworkadmin, app, common, directorywatcher, engine, core, metadata.repository, garbagecollector, ui.sdk.scheduler, repository, preferences, exemplarysetup, registry, updatechecker	By OSGI-based design
S_5	databinding.observable, core.net, core.filesystem, core.resource, core.runtime, core.variables, core.contenttype, RunLast, Debuglast, eof, StepOver, TerminateAndRelaunch, execute, AddBreakPoint, TogglebreakPoint, debug.commands.Execute, commands.Inspect, junitShortcut.rerunLast, gotoTest, ltk.ui.refactoring.commands. deleteResources, renameresources, moveResources, compare.selectPreviousChange	By refactoring, testing and debugging
S_6	delete, paste, copy, undo, text.goto.lineEnd, text.contentAssist.proposals, text.goto.wordNext, ui.file.save	By file modification
S_7	cdt.ui.editor, junitShortcut.rerunLast, gotoTest, delete, paste, copy, undo, text.goto.lineEnd, text.contentAssist.proposals, junitShortcut.debug, CompareWithWorkingCopyCommand, team.cvs.ui.CompareWithRevision, CompareWithLatestRevisionCommand	By reviewing and testing
S_8	synchronizeLast, TeamSynchronizingPerspective, synchronizeAll, applyPatch, create.refactoring.script, show.refactoring.history	By patch applying
S_9	mylyn.monitor.ui, mylyn.context.ui, mylyn.commons.ui, team.cvs.ui.commitAll, Commit, CompareWithRemote, Sync	By CVS committing
S_{10}	mylyn.monitor.ui, mylyn.bugzilla.core, mylyn.bugzilla.ui, team.cvs.ui.commitAll, Commit, CompareWithRemote, Sync	Updating issue tracking

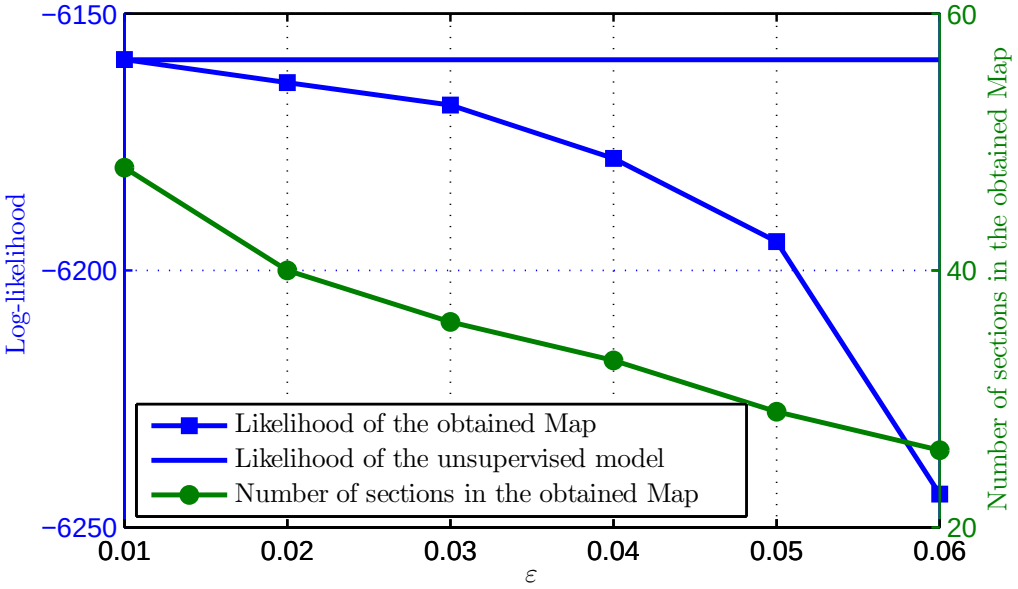

Figure 5.1: Likelihood and Number of Sections of the Discovered Map with respect to ε for the Eclipse Traces.

Observation 1. The developers' activities involve the usage of frameworks and plug-ins such as **Mylyn, Equinox, team/CVS, Junit**, built-in features of Eclipse such as **Eclipse Core, Debug** and API such as **ltk** (Language Toolkit). Figure 5.4 depicts the usage probabilities of these frameworks in each strategy. It is possible to observe the usage frequency of each framework/plugin/tools among the strategies. For instance, the usage frequency of *Mylyn* framework is 4 times among 10 strategies and its usage probabilities for strategy S_1 is 1 whereas for strategy S_5 it is 0. These values help detecting if any of these frameworks or tools is underused. For instance, one observation is that the refactoring tool **ltk** is underused since it has a low usage probability (see also observation 4).

Observation 2. The developers who start a development process choose one of the 4 first strategies *i.e.*, S_1, S_2, S_7, S_9. If they already have an ongoing program at hand, they tend to adopt either the strategy S_7: **by reviewing and testing** to **Improve the code** or the strategy S_9: **by CVS committing** to **Commit the code**. On the other side, the developers who start a new development choose either the S_2 strategy: **by regular programming activities** to **Initiate the development** or choose the strategy S_1: **by project tracking and team planning** to **Manage tasks** with probability 0.39 and 0.12 respectively. This observation suggests that the developers of this case study tend to start programming by building first the baselines for software architecture through dividing the programs into different modules/packages using **OSGI-based design**. After decomposing the soft-

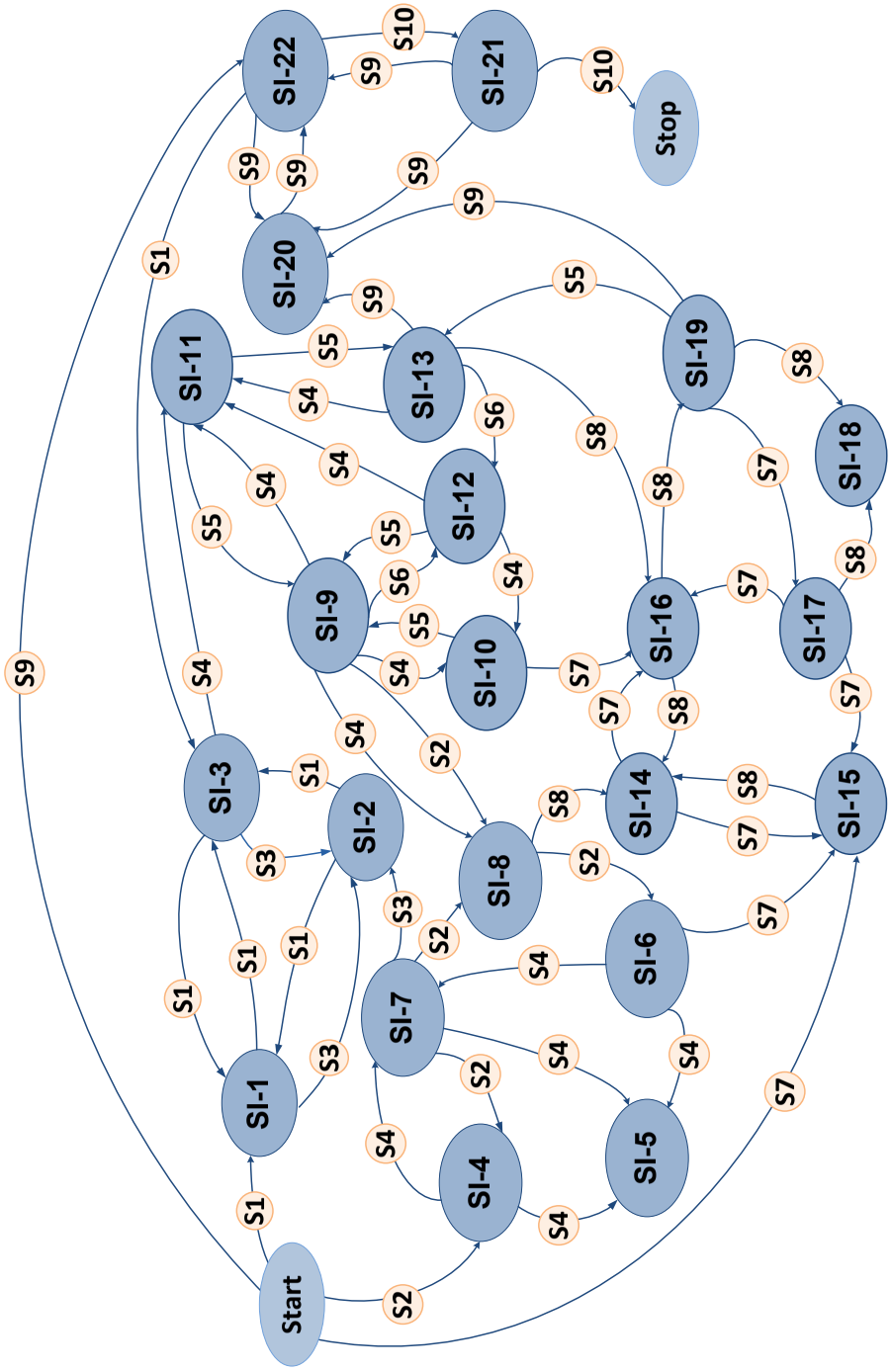

Figure 5.2: The obtained pseudo-Map for Eclipse UDC by MMM.

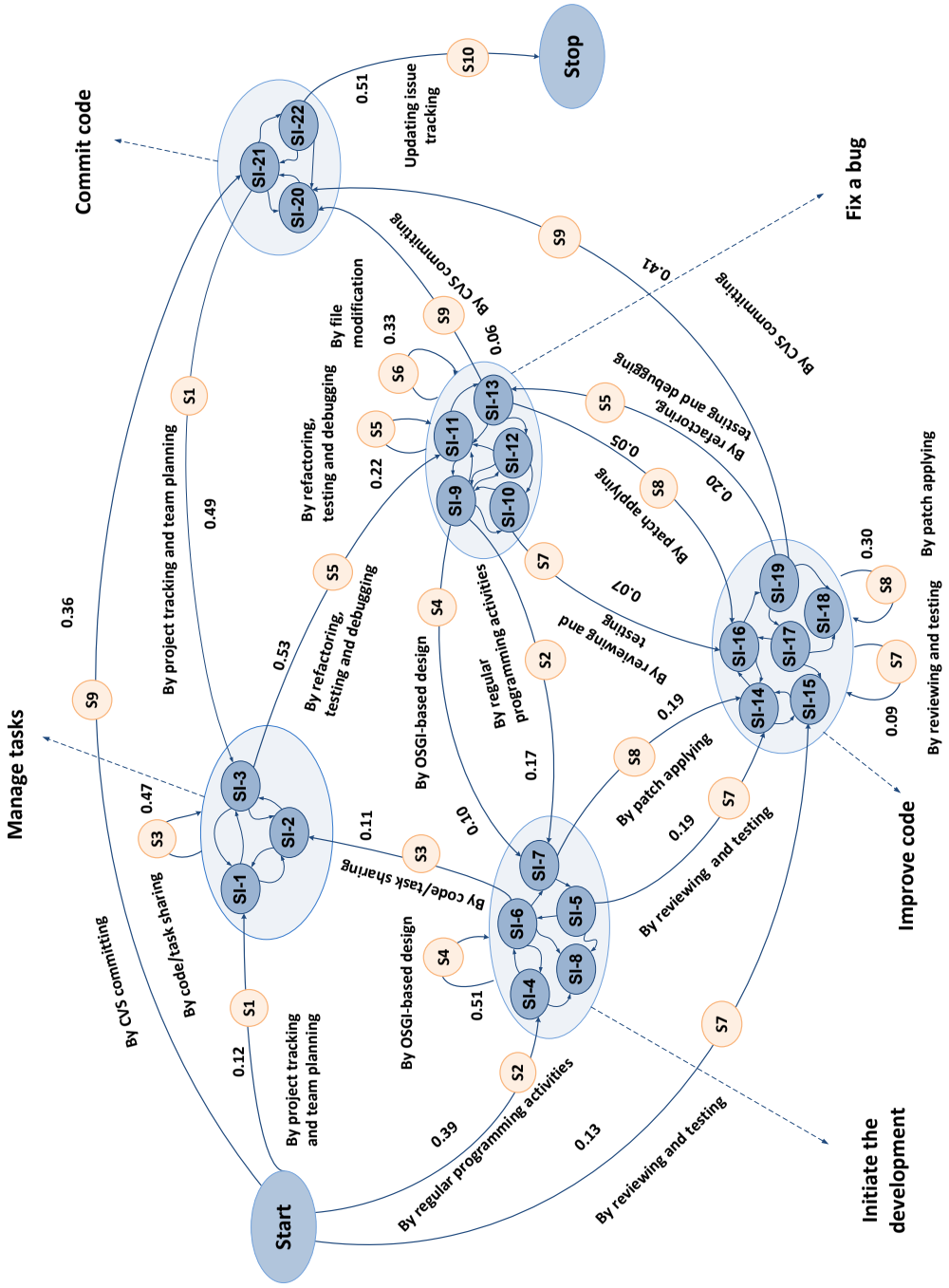

Figure 5.3: The obtained Map process model for Eclipse UDC by MMM.

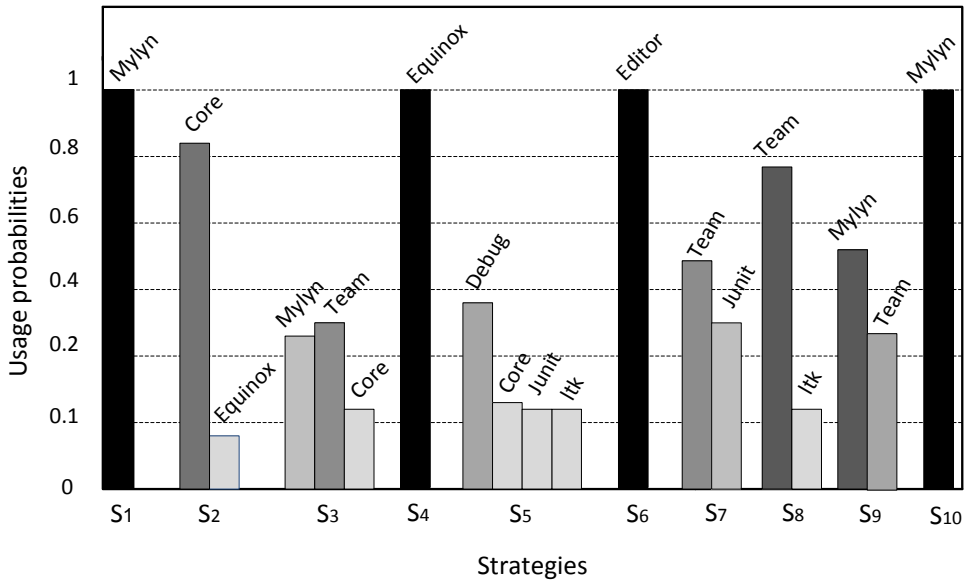

Figure 5.4: Usage probabilities of different Eclipse elements for each strategy.

ware into sub-modules, the developers utilize strategy S_3: **by code/task sharing** which involves mainly Mylyn framework (described in observation 3) which can be integrated with the Bugzilla bugtracker system and issue system. Therefore, tasks and the content of these tasks can be shared among developers. The high transition probability of this strategy means the developers tend to **share code/task** in order to fulfill **Manage tasks**.

Observation 3. The developers who have the intention to **Manage tasks** choose the strategies S_1 and S_3. Regarding Table 5.1, the activities related to these strategies involve Mylyn framework which is the task and application life-cycle management (ALM) framework for Eclipse. It helps the developers to work efficiently with many different tasks such as bugs, problem reports or new features. It monitors users' activities and preserve the context of the task-at-hand to focus the Eclipse UI on the related information. For instance, while working on a current task, if the developers have to work on another task, e.g. an occurred bug, Mylyn preserves the context of the current task. Thus, the developers can work on another task without losing the context of previous task. This procedure is discovered in the obtained Map of Eclipse. The activities of the developers while they **Manage tasks** are interrupted (*e.g.*, for an urgent bug) and they choose the strategy S_5 to **Fix a bug**. To switch to the previous task, they first commit and report the bug then they continue managing tasks. This means the Mylyn framework has reliable and relevant functionalities for developers and it is not underused since they work with it 47% (transition probability of 0.47) of their time and they use it frequently (Figure 5.4).

Observation 4. When the activities of the developers during **Manage tasks** are interrupted, they choose strategy S_5 to **Fix a bug**. The procedure of fixing a bug may involve refactoring existing code, writing unit tests, editing and modifying involved code and finally fixing the affected code. The strategy S_5: **by refactoring, testing and debugging** represents this procedure. This strategy is defined as a *best practice* to fix a bug, which means the developers who adopt this strategy respect the guidelines of software development. However, they perform this strategy with probability 0.22 whereas they prefer to **Fix a bug** by **file modification** with a higher transition probability (0.33). Moreover, Figure 5.4 shows a low usage frequency and usage probability for **ltk** (Language Toolkit) which is an API for automated refactoring in Eclipse-based IDEs. This means the developers prefer refactoring manually instead of using the refactoring and debugging tools of Eclipse provided for this purpose. This confirms some results of empirical research [Murphy-Hill 2008, Vakilian 2012, Fowler 1999, Xing 2006] that refactoring tools are seldom used and that they are not developers-friendly enough.

Observation 5. Bug fixing includes debugging and refactoring of different software modules. Once developers have fixed a set of bugs, they choose either strategy S_8: **by patch applying** with probability 0.05 or strategy S_7: **by reviewing and testing** with probability 0.07 to fulfill the intention of **Improve the code**. This observation means when debugging and refactoring have an impact on different parts of the program, the developers improve their code **by patch applying**. If debugging and refactoring change the program locally, they manually changes the code and send it for **reviewing and testing**. The probabilities describe that the developers tend to modify the code manually.

Observation 6. The Map can be used as a behavioral pattern to build recommender system for assisting developers in their daily development tasks. For example, this recommender system can suggest developers to avoid some problematic paths/strategies which might deviate from organizational best practices and to take some more effective paths/strategies. For instance, since the procedure of refactoring and debugging might affect the code in different parts of the program, it is recommended to check the coherence integrity of the program. To do so, after **fixing a bug**, the developers can be recommended to take the strategy S_4: **by OSGI-based design** to **Initiate the development**, to check if the refactoring and debugging have been impacted the **OSGI-based design** (*e.g.* if the dependencies between the modules are changed, etc).

5.2 Qualitative Evaluation of the Discovered Map

5.2.1 Context of the Experiment

To validate and evaluate the proposed method (MMM) with potential users, an experiment was conducted with 7 professionals in software development. This experiment consisted in individual qualitative interviews to collect their perception

Number	Average Age	Sex	Functions
7	33 (Min. 28 and Max. 38)	Male	1 CEO, 1 CTO, 5 Engineers of development

Table 5.3: Profile of the developers

and points of view on the Map process model obtained for Eclipse traces. This qualitative study aims at collecting different ideas and opinion, it is not made to quantify the use or usability of the tool or method. Table 5.3 summarizes the profile of the subjects. These subjects are developers from companies which have different activities such as mobile application, security, pharmacology software, aircraft software and Media wholesaler. All of them have more than 10 years seniority in development.

5.2.2 Description of the Protocol

At first, the definition of Map process models through an example is given to the subjects. Then the Map process model obtained by the MMM for Eclipse traces (Figure 5.3) is presented and explained to the subjects. Once they have become familiar with Map process model, the questionnaire given in Appendix B was followed to interview them. The questions are inspired by [Hug 2009]. The first part of the questionnaire is about their work habits. This allows better understanding their profile as well as the context in which the developers work and thereby better evaluate and analyze their answers to the questions about the Eclipse Map. The second part of the questionnaire is about Map process model, in order to understand their comprehension, perception and points of view regarding the Map process model. The final part of the questionnaire focuses on the Eclipse Map, described in Section 5.1. The goal of this part is to gather their opinions about the names of the strategies and the intentions, the relation between the activities and the discovered strategies, the transitions between the strategies and the topology of the Eclipse Map.

5.2.3 Results Analysis

5.2.3.1 Assessment of work habits

Regarding their development methods, 4 over 7 subjects said that they go through some development steps. However, 3 over 7 subjects said that they skip the development steps. *"We often pass through integration testing, unit testing and planning"* (subject 2), *"We do specifications phase after the development while it should be in other way, we try to correct this habit"* (subject 3), *"We pass through testing step"* (subject 5).

Only 3 over 7 subjects rigorously use design methods to develop software. *"We use UML diagrams because it is graphic and simple to use"* (subject 1), *"We use*

DesignPattern. First we analyze the requirements to use DesignPattern, we do not use UML diagrams. We use the design method of software development to reuse external library" (subject 1), *"I am inspired by DesignPattern for Eclipse but I do not really follow it. It gives me some ideas. I do not use design methods to develop software because either they are not formal or they are not adapted to the requirements"* (subject 3), *"We do not use them because it is very difficult to apply the pure method on practice, for example the UML diagrams are very difficult to implement while we can easily do the things manually"* (subject 5), *"We organize the development of our software with SCRUM, by dividing it to the modules, then we prioritize the modules from the most important to the less important one, then we start the development phase"* (subject 6), *"I used to use DesignPattern but now, I do not use any method"* (subject 7).

The majority of subjects (5 over 7 subjects) monitor their methods. *"We use sub-version tools to monitor software evolution, this allows stepping back to chose the best version of code, but we do not have a specific monitoring*" (subject 1), *"We have a meeting once per week and we discuss with each other by sharing what we are doing*" (subject 2), *"We have a deposit system, it means for each step of development, we commit our code. Therefore we can see the different modifications regarding ancient version and we ensure that the ancient functionally are preserved by predefined tests or functionality tests. We use SVN for this"* (subject 3), *"We only monitor the quality of our code with JSLint"* (subject 4), *"Since we work with SCRUM, before the development we estimate the complexity of the each module which corresponds to the development time, and we will check the real time that the module took after development. By comparing these times we can have a feedback on the real time that it takes. We also test each module to ensure that the functionality has been correctly implemented"* (subject 6).

All subjects (7 over 7 subjects) said that they do not respect design methodology to the letter and they adapt the methods to their requirements.

Only 1 over 7 subjects said he presents his methods to other people. *"In our company, we try to have the same method to be consistent with each other. We try to have the certification of ISO 9001 for quality, therefore we try to construct [our code] well to have it"* (subject 3).

Only 3 over 7 subjects said they document their method. *"We try to comment our code, we try also to document the report of specifications in Word and we schematize it in Powerpoint"* (subject 3), *"We document the method first manually then we use Oxygen tool, we have both internal and external documentations*" (subject 5), *"I use Github for documentation"* (subject 6).

5.2.3.2 Assessment of Map process model

Regarding process modeling, 5 over 7 subjects said they have never heard about it. However all subjects (7 over 7 subjects) believe that Map process model is easy to understand and it is fairly intuitive. All subject believe that Map process model can be learned quickly (around one hour).

The first impression of the developers on Map process model is generally counter-intuitive. *"We are a little company and it is particularly interesting for large companies, but Map process model can be useful for optimization of methods*" (subject 1), *"I do not know what it offers to me [Map process model]"* (subject 3), *"I cannot see how Map process model can help us model our process"* (subject 4), *"I cannot see immediately the interest of Map process model"* (subject 7).

After some explanations, they understood the benefits of Map process model. *"I think by modeling [developers] with this kind of model [Map process model] it is possible to extract the best path which are the best practice, this allows realizing that the best path is not always what is prescribed. It is a significant gain of time*" (subject 1), *"It allows avoiding the oversights of steps. We can skip some steps but it should be voluntary. It can highlight some tools we did not know, or to find them out efficiently"* (subject 2), *"When we try to execute a task, since we always do it in the same manner, Map allows extracting the things"* (subjects 3), *"Since I am bad at organizing my tasks, [Map process model] can be help me to be organized"* (subject 6).

Regarding the disadvantages of Map process model, all subject said one disadvantage is that it should be applied on several developers to have enough data and then extract a more interesting good behavioral pattern. Other disadvantages were also expressed. *"The first disadvantage for me is following the succession of the steps too automatically which may cause wasting time, for example, if we want to solve bugs with another solutions by skipping some steps*" (subject 2), *"I cannot see the time on the transitions. It would be interesting to know how much time is necessary for a transition"* (subject 3).

4 over 7 subjects said they are ready to recommend using Map process model for large companies. One subject emphasizes that: *"It must be transparent for user, it means it should not create an overhead for user, for example, it should run on a plug-in"* (subject 3).

5.2.3.3 Assessment of The Eclipse Map

As a first impression, all subjects (7 over 7 subjects) found that the Eclipse Map is interesting, since it generally illustrates the typical developers' behaviors and their daily habits.

All subjects (7 over 7 subjects) believe that the Eclipse Map reflects most of typical developers' behaviors. *"You cannot log everything that happens during process development, for example when you log only Eclipse, when a developer jump to another software you do not have access to the latter logs, it is more interesting if you can log every activity of the developers*" (subject 1), *"The intentions are very significant for me, I identify right away my daily tasks in the Map, particularly improve code and fix a bug. Regarding the strategies, I generally find them relevant with my habits, but I would like to add some strategies such as the interactions with users, because when we fix a bug we communicate with the users by email"* (subject 2), *"Generally I can see our daily tasks. I cannot see regression test and conformity,*

i.e. to test if something we have done before still works. But I do not know if I would still agree with these concepts on a more detailed level" (subject 3), *"What I cannot see is consulting the manual. Personally I check the documentation a lot, another thing is testing, we test the code permanently. I cannot see this in the Map directly*"(subject 4), *"I can find out all the steps in the development, the transitions between intentions make definitely sense, I can even trace my usual path but maybe not with the same probabilities"* (subject 6).

All subjects (7 over 7 subjects) found that the Eclipse Map can help developers who are already initiated to Eclipse. *"I think that the Eclipse Map can help to train the beginners of Eclipse platform, some strategies seem definitely helpful to initiate the development process, since it is interesting to have the same habits work in the same company*" (subject 1), *"It is helpful for somebody who is novice in tasks organization but more experimented on Eclipse*" (subject 2), *"The Map allows developers comparing them-self to others*" (subject 3), *"I think it is helpful to formalize the best practice of developers or monitoring the activities of a given developer"* (subject 4), *"The Map can be helpful to assist any developer (Eclipse or not)"* (subject 5), *"I do not think a novice developer can use the Map. It is for a non-beginner developer"* (subjects 7).

4 over 7 subjects said that the Eclipse Map can be useful to improve the quality of the development method. *"It can be useful by proposing the best path to choose which is not already mentioned in a manual*" (subject 1), *"It can help avoiding the oversights or to have the new ways to do something, but I think this highly depends on company dimension, if it is automated we can use it to improve the development process"* (subject 2), *"The Map can be useful if it is automated, it can be useful for real-time recommendation*" (subject 3), *"I think the Map can help avoiding oversights of some tasks such as test and optimization phases"* (subject 5), *"If the really Map reflects the best practice the developers, we can use it to improve our code otherwise it would be difficult"* (subject 6).

All subjects generally found the inferred name of strategies and intentions relevant. However, they suggested some names for some strategies. *"For a developer, the names of intentions and strategies definitely make sense. The activities seem grouped into the right classes of strategies*" (subject 1), *"The names of the intentions are general enough which helps understanding, I would like to add some strategies such as interactions with client*" (subject 2), *"The names of the intentions and the strategies seem consistent. I would like to add the strategy of by comparing for the intention improve code. And for the same intention, I do not agree with by patch applying, I would put something like by refactoring. I do not know for the first strategy, and I agree with the names of the other strategies"* (subject 3), *"The names of the strategies and the intentions seem consistent to me"* (subject 6).

4 over 7 subjects found it is interesting to have the transitions probabilities on the strategies. *"[observation 1] I agree with this because it allows having some feedback about the usage probabilities, for example to find out if the tools are adapted or not, it is indirect usage [of the Map]. For the observation 5, I think the intention can be refined. If we have a local bug or a bug that can impact the software, maybe*

we do not have the same probability" (subject 2), *"I agree with observation 2, since the developers tend to start the development by dividing it in different modules. The observation 3 seems consistent too. For observation 5, one should change the name of by patch applying to by refactoring"* (subject 3), *"The probabilities help finding the best path on the Map to improve the development"*(subject 4), *"I think the name of the strategies are very close to the activities. For me, a strategy is an intellectual concept. I would like to see an applicative sense in the name of the strategies"* (subject 6).

Almost all subjects found that it is highly necessary to have the detailed level of the Map. *"At first, it is interesting to have a generic view but then it is more interesting to focus on some parts [intentions], particularly if it is possible to navigate through the Map via a tool*" (subject 1), *"I think we can refine the intention fix a bug into two sub-intentions of fix a major bug and fix a minor bug. Then we do not have the same strategies, we will have something like safe-strategy and testing*" (subject 2), *"I wish I could see the more detailed level. That would make more sense to me"* (subject 3), *"In this level of the Map, I cannot find out some intentions such as initiate the functionality. For me, it could be somewhere in initiate the development, therefore I think if we have a more detailed level of the Map, we can find some sub-intentions in the intentions"* (subject 4), *"For us, it is important to have the clients feedback to fix a bug or improve it, but this part does not exist on the Map"* (subject 5).

All subjects found that the Eclipse Map can be used to formalize problems. Three of them precised their thoughts. *"I do not know if the Map can be useful to formalize problems, I can see the path that lead to fix a bug but I cannot see how the bug is created*" (subject 1), "*For me, managing a task is difficult and the Map may guide me to overcome this problem*" (subject 2), *"The Map could help me for committing the code that I usually forget"* (subject 6).

5.2.3.4 Synthesis

These qualitative interviews with developers helped to highlight several points:

- The topology of the Eclipse Map is consistent with the experiences of the developers. They found it relevant with their daily activities.
- The names of the strategies and the intentions have generally been validated by the developers, although some of them have suggestions to improve them.
- They would need an effective tool for modeling intentional processes automatically. They wish they could have an interactive model in each step of development. This should not create an overhead for the developers.
- The level of abstraction (sub-intentions) should be available for each intention since the developers need to see what are the components of intentions. This is interesting to better understand the deep nature of a process.

- Map process models are particularly useful to model the behaviors of many developers in large companies.
- The developers generally pass through the step of testing or they do it an informal way. This may be the reason why in the Eclipse Map, this step is not very highlighted.
- The obtained Map can help developers avoiding the oversights in each steps of development process.
- Many of the developers mentioned that they would like to see the interactions and feedback of the clients. Therefore, it would be interesting to trace these interactions along with the development traces and generate a Map process model from both of these traces.

5.3 Threats to Validity

There are four main issues that threats the validity of the proposed approach.

First, the mined Map may suffer from under-fitting problems if the number of activities traces used to estimate the parameters of the HMM is not high enough. Indeed, these traces have to capture all the possible behaviors while enacting the process under study to produce an accurate Map.

Second, the BWA requires an important number of iterations to converge to a result. For instance, it converges at $20,237$ learning iterations for the Eclipse case study. Moreover, it cannot always be guaranteed to converge to the global maximum likelihood.

Third, although the MMM automatically discovers the topology of the Map process model, the names of strategies and intentions are still inferred semi-automatically. The manual part of this naming procedure introduces a human bias. However, the logical relations between activities, strategies and intentions established in the obtained Map could be exploited to build an ontology to fully automate the process of inferring the names of strategies and intentions.

Fourth, the M_1M_0 topology chosen for the HMM is the most sophisticated topology allowing the use of an algorithm such as BWA or equivalent algorithms. More complex topologies may actually be more appropriate to model some processes. However, there is no known algorithm to estimate the parameters of these topologies, as BWA does for M_1M_0. For this reason, the scope of MMM is limited to the M_1M_0 topology.

5.4 Conclusion

This chapter presented the application of MMM on a large-scale and real world case study. The results are convincing from both theoretical and practical points of view. As a criterion for evaluation of the method, the likelihood that is a well-known

criterion for statistical model is used. The results demonstrate that the likelihood of the unsupervised parameters is higher than the discovered Map. This phenomenon demonstrates that the results directly obtained by BWA have higher probabilities to generate a model that the best corresponds to the observed traces (*i.e.*, what really happened during process enactment). However, the likelihood of the discovered Map is lower than the model parameters generated by BWA due to the constraints of the structure imposed by the topology of a Map. The observations obtained from the discovered Map are useful for an analyst, a software designer to improve the software usability [Bass 2003]. The Map can help the novice or unfamiliar users learning system features by using the Map obtained from previous development process as a guideline. For instance, when their intentions are known, they can be recommended which strategies and activities that might be useful to fulfill their intentions. Since the intentional topology of a Map makes it user-friendly, it can help software designers designing systems that enable users to be more efficient in their operations. This can be done by adapting the system to the users' needs or by assisting the users step by step during the process enactment. Using a Map increases also the users' confidence and satisfaction in the enactment of a process. All these points contribute to improving the usability of the software products or information systems.

Chapter 6

Map Miner Tool

Contents

This chapter presents the Map Miner tool, which automates different modules of MMM :These modules are (i) estimating the emission and transition matrices by unsupervised learning, (ii) discovering the pseudo-Map by Deep Miner algorithm, and (iii) discovering the Map process model by Map Miner algorithm. Note that Map Miner tool does not automate neither inferring the names of strategies and intentions nor generating the graphic Map process models (*e.g.*, with nodes and arcs). The Maps will be generated as the text and they should be design by the user.

Map Miner tool allows discovering Map process model from traces using unsupervised learning. Map Miner tool allows visualizing the Map process models in different levels of abstraction. It is easy to manipulate and to understand for an end-user.

The Map Miner tool takes as inputs an trace comprising timestamped users' activities. Then it loads the data into the embedded database. Several parameters must be adjusted to run the Map Miner tool, such as strategies number, the threshold ε for obtaining pseudo-Map and the number of high-level intentions for obtaining the Map process model. Map Miner tool is developed in large part in Java programming language.

6.1 Format of Input Files for Map Miner Tool

Map Miner tool can take as inputs either an Excel spreadsheet files or data loaded already in a external database since external database can directly loaded into the Map Miner tool database. However, in both cases the files must be ordered as depicted on Figure 6.1.

The first column contains the UserID, which can be any kind of identifier such as numbers or names. The second column contains the timestamps, which determine the time and date of activities execution. The third column contains the activities that users performed while enacting a given process.

The order of the columns must be respected but the order of rows is not important since Map Miner tool orders them by timestamps and users ID. It is important to note that Map Miner tool is sensitive to inputs types, which means it does not support a cell of Excel file containing the value "NULL", empty, with quotes, space or any other kind of characters except numbers and letters.

	A	B	C	D
1	UserID	Timestamps	Activities	
6324	12515	1,277E+12	org.eclipse.equinox.p2.metadata.repository	
6325	12515	1,277E+12	org.eclipse.equinox.p2.reconciler.dropins	
6326	12515	1,277E+12	org.eclipse.equinox.p2.repository	
6327	12515	1,277E+12	org.eclipse.equinox.p2.ui.sdk.scheduler	
6328	12515	1,278E+12	org.eclipse.equinox.p2.updatechecker	
6329	12515	1,278E+12	org.eclipse.equinox.preferences	
8953	15351	1,277E+12	org.eclipse.equinox.common	
8954	15351	1,277E+12	org.eclipse.equinox.ds	
14617	12515	1,277E+12	org.eclipse.jdt.core	
14618	12515	1,277E+12	org.eclipse.jdt.core.manipulation	
14619	12515	1,277E+12	org.eclipse.jdt.launching	
14620	12515	1,277E+12	org.eclipse.jdt.core	

Figure 6.1: A fragment of Excel file that can be used in Map Miner tool

6.2 Map Miner Tool Interface

Map Miner tool can be executed directly in command-line. To make it more interactive for end-user, a graphical user interface is also designed (see Figure 6.3). This interface allows opening the file containing traces by selecting the tab "File" in the left corner of the interface windows. Then the desired file with the extension of *.xls can be chosen.

As shown in the figures, a user can adjust the parameters of Map Miner. These parameters are explained in the next section.

←T→	id	time	transition	user
Modifier Copier Effacer	1	1276575174851	org.eclipse.cdt.ui.editor	2106712.0
Modifier Copier Effacer	2	1276748577482	org.eclipse.compare	63897.0
Modifier Copier Effacer	3	1276748577493	org.eclipse.core.	63897.0
Modifier Copier Effacer	4	1276748577502	org.eclipse.core.	63897.0
Modifier Copier Effacer	5	1276748577513	org.eclipse.core.	63897.0
Modifier Copier Effacer	6	1276748577559	org.eclipse.core.jobs	63897.0
Modifier Copier Effacer	7	1276748577560	org.eclipse.core.	63897.0
Modifier Copier Effacer	8	1276748577574	org.eclipse.core.	63897.0
Modifier Copier Effacer	9	1276748577575	org.eclipse.core.runtime	63897.0
Modifier Copier Effacer	10	1276748577584	org.eclipse.core.runtime	63897.0
Modifier Copier Effacer	11	1276748577585	org.eclipse.core.	63897.0
Modifier Copier Effacer	12	1276748577596	org.eclipse.debug.	63897.0
Modifier Copier Effacer	13	1276748577597	org.eclipse.ecf.	63897.0
Modifier Copier Effacer	14	1277090161618	org.eclipse.epp.usagedata.	34393.0

Figure 6.2: A fragment of the embedded database used in Map Miner tool

6.3 Inputs Parameters

There are several parameters to be adjusted for both generation of pseudo-Map and Map process models. This makes Map Miner very flexible and user-friendly since the user can customized the parameters according to the context at hand or to obtain some tailored results. The parameters that can be adjusted in Map Miner are as follows:

- The value of ε (minimum probability): as described in chapter 4, when ε is close to 0, almost all the transitions are present in the discovered Map. However, when ε increases, the number of sections, as well as the likelihood of the discovered Map, decrease. The Map gets more easily understandable by humans but it is not as accurate in terms of transition. Therefore, the value of ε has to be set regarding the desirable accuracy of the Map and its understanding. For instance, we adjust the minimum probability to 0.1, which means all the transitions lower than this value will not be taken into account by Map Miner tool.

- Number of strategies: as found in Section 4.3.2.4, the number of strategies in MMM framework was obtained by a heuristic method. This method is a brute-force method since it consists in generating all models with different numbers of strategies and observing the associated emission matrices until that the number of different strategies obtained in the emission matrices reaches a

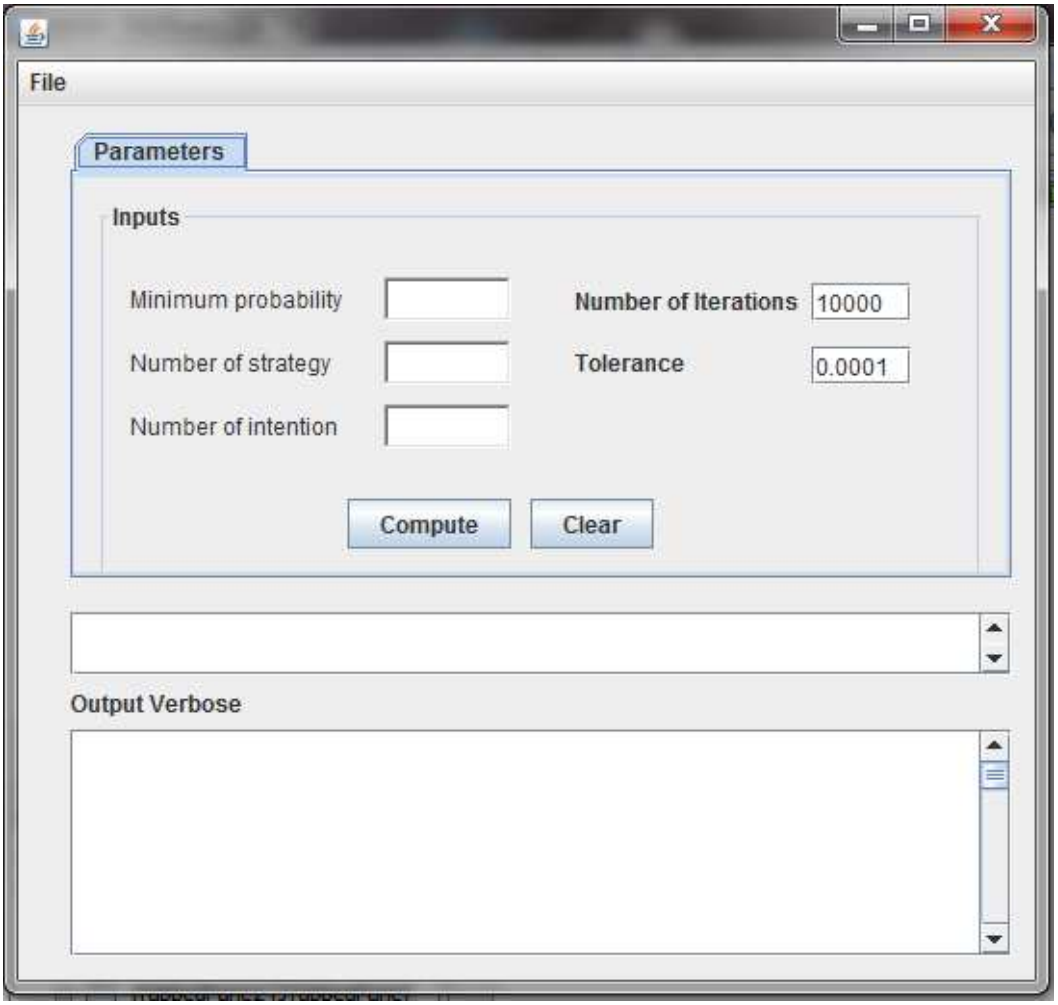

Figure 6.3: An overview of Map Miner Tool

threshold. Therefore, although this heuristic method is more adapted to the context of each trace, however it is a time-consuming method. For this reason, in Map Miner tool users can determine the number of strategies manually. When the number of possible strategies is too high, the BWA produces an emission matrix with several identical strategies. Consequently, the right number of strategies corresponds to the observed threshold. For instance, if the number of strategies is adjusted to 10, and the discovered Map shows two identical sections, this means the right number of strategies is 9.

- Number of high-level intentions: this parameter needs to be adjusted to obtain a Map process model. The **Start** and **Stop** intentions are excluded from it. For instance, if one adjusts the number of high-level intentions as 4, the resulted Map will contains 6 intentions in total, *i.e.*, the **Start** and **Stop** intentions and 4 high-level intentions.

- Number of iterations: The convergence of the BWA used in Map Miner depends on the initialization of the matrices **T** and **E**. Therefore, the higher the value of the number of iterations is, the BWA converges more to a local maximum. The user can set up this value to a lower value to stop more quickly the algorithm.

- Tolerance: The number of iterations of the BWA depends on a stopping criterion. This criterion is the minimal difference between the likelihood of the estimated HMM at two consecutive iteration of BWA.

  Note that the number of iterations and tolerance are set up in advance. However, one can change it regarding the expected results.

Figure 6.4 illustrates an example of adjustment of the parameters in the Map Miner tool interface.

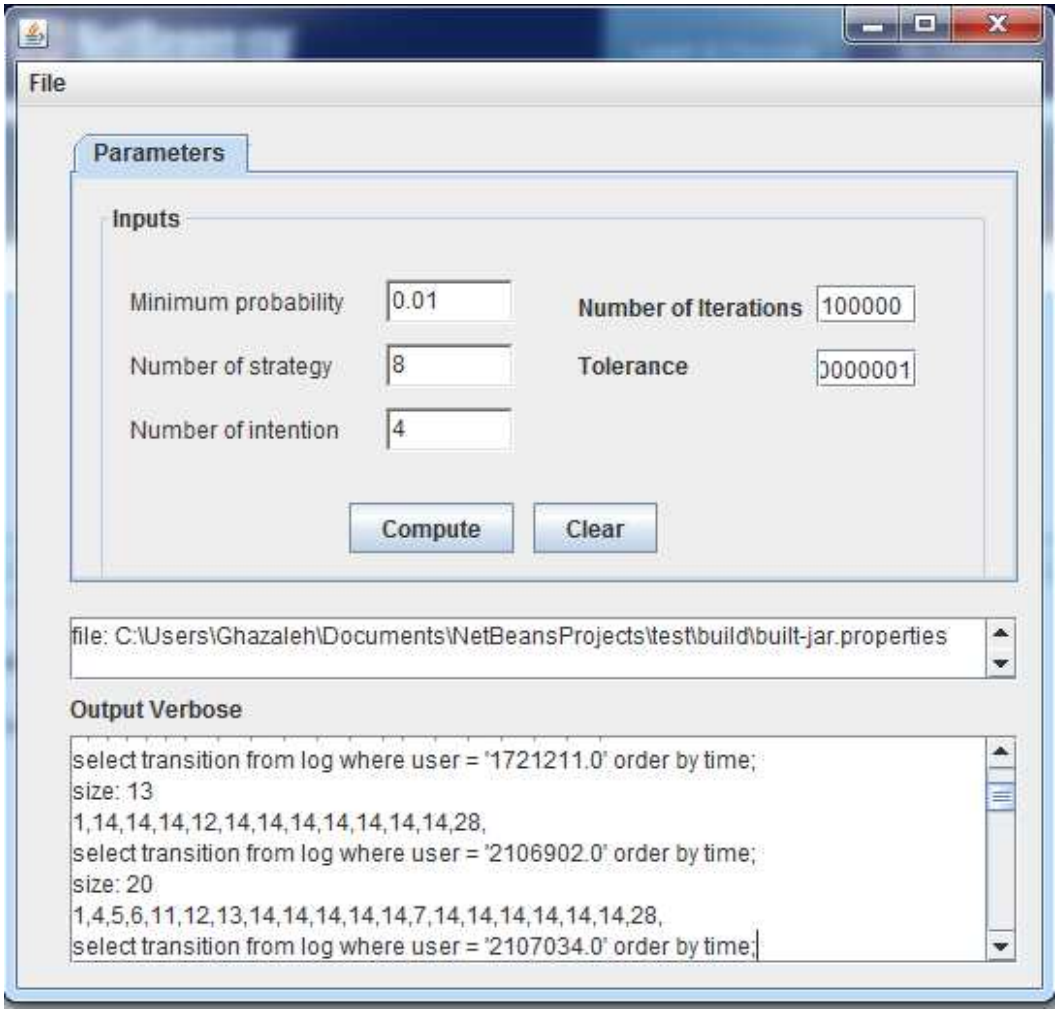

Figure 6.4: Verbose outputs of Map Miner tool

6.4 Outputs of Map Miner Tool

The outputs of Map Miner tool appear: (1) on the bottom of interface windows: these outputs are verbose and contain all the information about the dataset, such as number of users, traces, time, etc and also all outputs of Map Miner Method, such as emission and transition matrices, etc(see Figure 6.4); and (2) into a new windows: these outputs report only the necessary information to construct pseudo-Map and Map process models as well as the activities related to discovered strategies (see Figures 6.5, 6.7, 6.8).

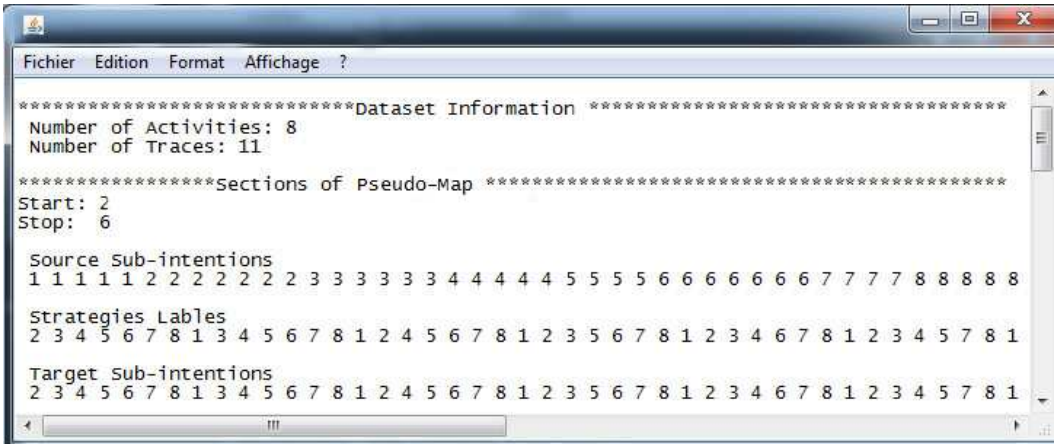

Figure 6.5: Traces information and section of pseudo-Map

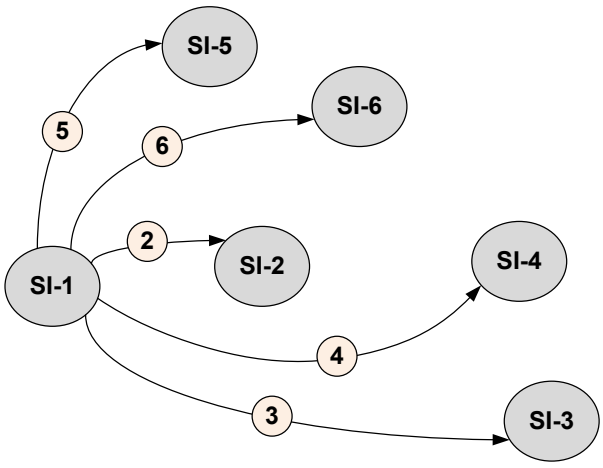

Figure 6.6: An example of the construction of pseudo-Map

Map Miner generates pseudo-Maps at different level of accuracy depending on

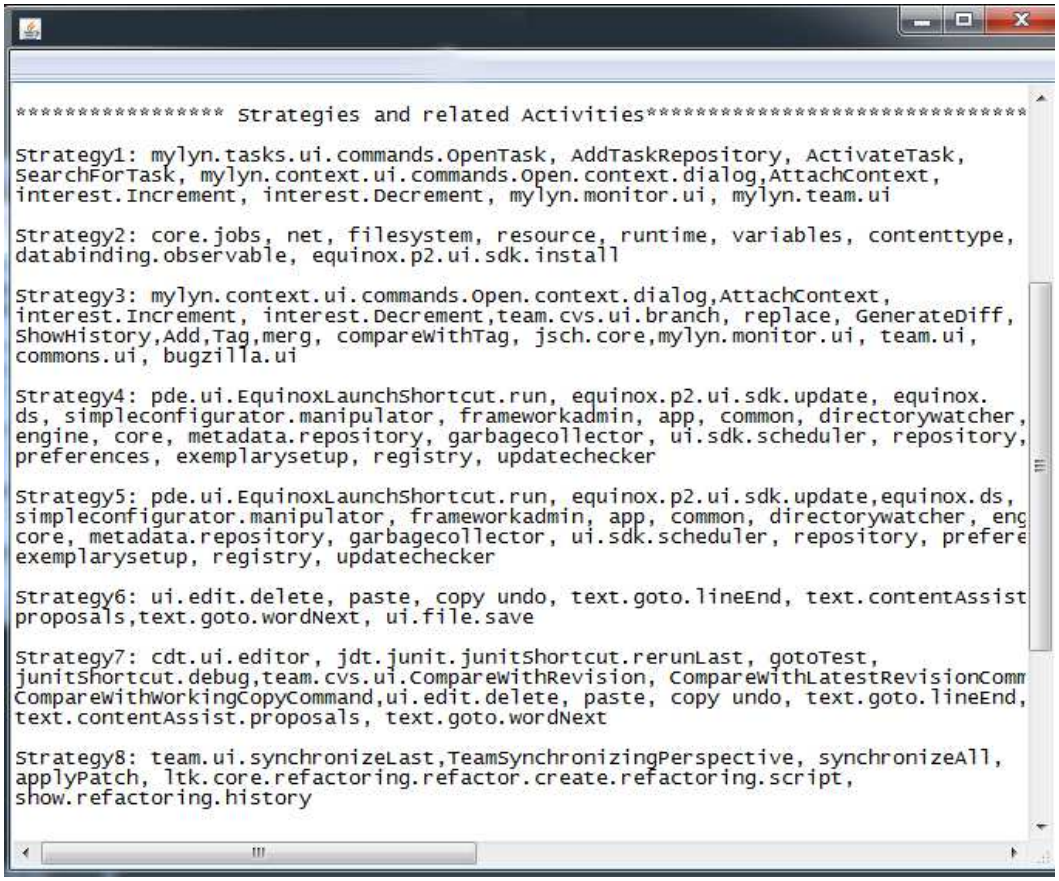

Figure 6.7: Discovered strategies and related activities

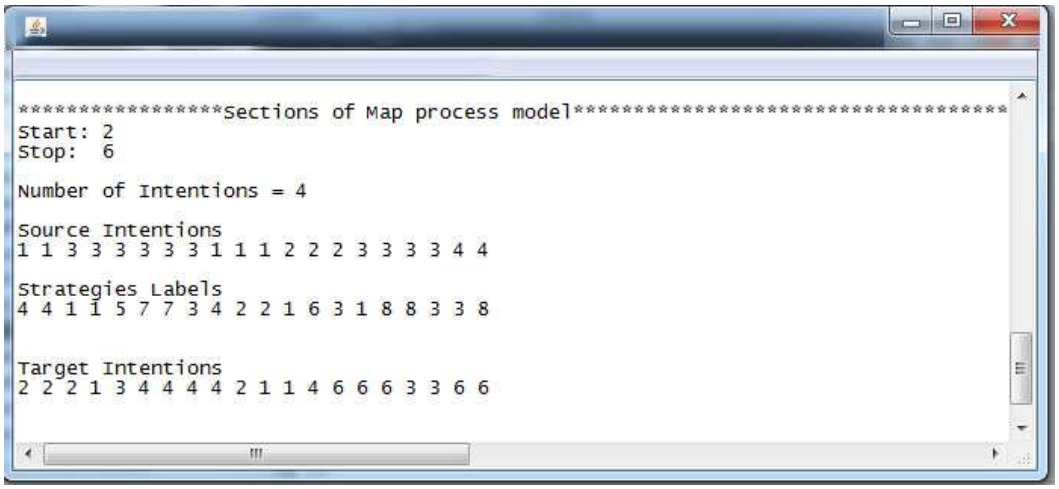

Figure 6.8: Sections of Map process model

the number of strategies and the value of minimum probability (ε). The closer to 0 is ε, the higher the number of sections on the Map are. Another output of Map Miner is a Map process model with high-level intentions. To obtain a Map process model, first there has to be a pseudo-Map. Indeed, Map Miner takes the sections generated out of the pseudo-Map and the number of intentions adjusted by the user.

In Figure 6.5, pseudo-Map is presented by some index for source sub-intentions, strategies, and target sub-intentions. The index of *Start* and *Stop* are also indicated. The results discovered by Map Miner tool reads as follows: the elements of source sub-intentions must be related to the elements of target sub-intentions through the corresponding strategies. For instance, the first element of source sub-intention "1" must be related to the first element of target sub-intention "2" through strategy "2" or the second element of source sub-intention "1" must be related to the second element of target sub-intention "3" through strategy "3", and so forth. These results can be designed by user in a graphical way. Figure 6.6 depicts a graphical example for the first five sections discovered in the pseudo-Map (Figure 6.5).

In the same manner, the Map process model can be read (see Figure 6.8). For instance, the first element of source intention "2" must be related to the first element of target intention "1" through strategy "4" or the second element of source intention "1" must be related to the second element of target intention "1" through strategy "4", and so forth. Note that the index of *Start* and *Stop* remain the same as before. Figure 6.9 depicts a graphical example for the first five sections discovered in the Map process model (Figure 6.8).

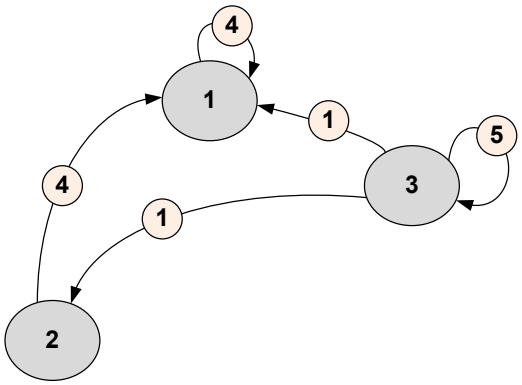

Figure 6.9: An example of the construction of Map process model

6.5 The Programming Languages of Map Miner Tool

MMM has been initially developed in *MATLAB*. All the experiments and case studies of chapters 5 and 4 have been treated in *MATLAB*.

MATLAB is a computer program that provides the user with a convenient environment for performing many types of calculations. Besides, an effective, quick and easy way to solve differential equations. Moreover, it allows easy numerical calculation and visualization of the results without advanced and time consuming programming. *MATLAB* is an interpreted language. This implies that the source code is not compiled but interpreted on the fly: it can be slow, especially when bad programming practices are applied.

Nevertheless, the ultimate goal of MMM is to be a tool available for the community. This allows, on the one hand, checking its robustness, relevance and reliability; on the other hand, it enables users to study processes under an intentional angle.

In this perspective, MMM is also implemented in Java programming language (see Appendix A). The advantages of Java are as follows [IBM 2014]:

- Java is easy to learn. Java was designed to be easy to use and is therefore easier to write, compile, debug, and learn than other programming languages. This characteristic makes the MMM code easy to understand and expendable in the future.

- Java is object-oriented. This allows creating modular programs and reusable code. Thereby the MMM code is reusable and improvable to be adapted to the context of projects at hand.

- Java is platform-independent. One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels. This allows Map Miner tool be pluggable to different platform such as ProM [EUT 2013].

Because of Java's robustness, ease of use, cross-platform capabilities and security features, it has chosen for the implementation of Map Miner tool.

6.6 Limitations of Map Miner Tool

There are some limitations that hinder the Map process Models generated by Map Miner.

First, the number of activities in the traces has to be high enough to capture all the possible behaviors while enacting the process under study. This allows producing an accurate Map, which does not suffer from underfitting problems. In other words, the number of unique activities in traces has to be proportional to the number of all activities in traces.

Second, for complex datasets, Map Miner requires an important number of iterations to converge to a result. This can be time-consuming in the case of complex dataset. For instance, it took about 30 minutes for the Eclipse UDC. However, it is

possible to set up the number of iterations to a smaller value to decrease the time of algorithm execution, which leads to poor results.

Third, although the Map Miner automatically discovers the topology of the Map process model, the names of strategies and intentions are still inferred manually. The manual part of this naming procedure introduces a human bias. However, the logical relations between activities, strategies and intentions established in the obtained Map could be exploited to build ontologies to fully automate the process of inferring the names of strategies and intentions.

Fourth, although the parameters of Map Miner allow customizing the discovered Map, the adjustment of these parameters needs some expertise.

Chapter 7

Conclusions and Open Issues

Contents

7.1 Conclusions

This thesis proposed a new vision of process mining by focusing on the intentional angle. This means discovering the actual processes from event logs and modeling them using intentional process model (Map). This method is called Map Miner Method (MMM), which models processes automatically in terms of users' intentions and strategies. Intention-oriented process models permit modeling humans' cognition operators, *i.e.*, thinking, reasoning, deciding creative process, which is not possible with activity-oriented process models. The discovered Map process model answers to the problems of *why a process is enacted*. Although MMM is not designed to answer to the process mining problems, the case study presented in chapter 5 shows that the discovered Map process model is not hampered by the same problems identified in process mining, such as hidden tasks, duplicate tasks or the problem of loops. This is due to the fact that Map process models are flexible and activities are of less importance with a representation on a higher level, *i.e.*, intention level. On the other hand, Map process model can handle the loops as its sections can be enacted several times, until the desired intention is achieved.

Apart from Map process models discovering, the results of this thesis will enable checking the conformance, improving the models and providing the recommendations :

- Conformance checking: formalizing the models followed in practice allows comparing them to the original models prescribed by the organization and possibly adapt and improve prescribed models to actual practice and finally provide recommendations for stakeholders.
- Enhancement: discovering Maps are useful for an analyst, a software designer to improve the product or method usability. Since the intentional topology of a Map makes it user-friendly, it can help software designers designing systems that enable users to be more efficient in their operations.

- Recommendation: provide better guidance for computer systems users, through recommender systems for assisting them in their daily development tasks. This guideline is adapted to the users' context taking into account the experiences of previous users and actual users' intentions. The Map can also help the novice or unfamiliar users learning system features by using the Map obtained from previous development process as a guideline.

In this last chapter, first, the contributions of this thesis will be summarized by mapping them to the research questions raised in the introduction. Then, the limitations of the thesis will be pointed out. Finally, the outlook of the thesis for future research will be indicated.

- Q_1 How can dependencies within activities be discovered to estimate the strategies? Event logs reflect which activities are really performed by users while enacting a process. The dependencies within activities can express different ways of process enactment.
  - $H_{1.1}$ It is possible to discover the dependencies within activities from event logs.
  - $H_{1.2}$ A strategy consists in a set of activities.

  The dependencies within activities logs are discovered and thereby the users' strategies are estimated. Discovering the strategies allows understanding the different ways of working to achieve a goal (the intention). The strategies are estimated in both supervised and unsupervised learning.

  Supervised and unsupervised learning approaches of HMMs are compared from both theoretical and practical points of view, to discover process models. This allows determining which learning approach is more appropriate to the situation at hand. To run supervised learning it is necessary to have the traces of strategies related to the traces of activities. Therefore, it is a costly technique, since the users' should label their activities. In contrast, unsupervised learning can be run without such information. In fact, the cost of labeling the activities by the strategies for supervised learning is quite high as it involves the users' commitment at each step of the process. In comparison, unsupervised learning requires no users' effort. This characteristic makes unsupervised learning a method easy-to-use. Therefore, only unsupervised learning can be used to automate MMM. In addition, regarding theoretical and practical results, the likelihood of unsupervised learning is higher than supervised learning. This means that the results obtained by unsupervised learning reflect better the real users' behaviors (*i.e.*, users' strategies and intentions) than supervised learning. For these reasons, unsupervised learning of HMM is highly recommended to estimate users' strategies during the enactment of a process. However, supervised learning is useful to validate the hypothesis in practical manner. The supervised learning also allows checking the alignment between the prescribed models and the discovered models.

- Q_2 How can the intentions be identified from the estimated strategies? Once the strategies are estimated, the link between these strategies and intentions have to be discovered.
  - H_2 A strategy is followed to fulfill a given intention.

  For the estimated strategies the intentions are discovered regarding the Map formalism constraints. Indeed, in this formalism, the strategies lead to the nodes, which are the intentions. Therefore, once the strategies are discovered, it is possible to detect where they lead, *i.e.*, the intentions. This allows reconstructing the topology of a Map process model.

- Q_3 How can the processes be modeled at different levels of abstraction? The nature of granularity that is needed to model a process can be defined regarding the situation at hand.
  - H_3 Clustering techniques can be used to abstract the discovered intentions.

  Deep Miner algorithm is developed to discover the pseudo-Map from traces. In this perspective, a new metric of fitness-precision is proposed to avoid the overfitting or underfitting problems. Therefore, the Map process models are obtained by optimizing the precision-fitness metric, which guarantees discovering a model that fits the actual process. This kind of Map process model provides different levels of details for the actual process (regarding the chosen parameters). This is helpful for analysts or process designers to visualize and to analyze the processes in various aspects.

  Map Miner algorithm is developed to generate the Map process models from pseudo-Maps. This algorithm uses the clustering technique to group low-level intentions into high-level intentions. The low-level intentions are represented in a space in which they can be classified into clusters. This allows visualizing the processes in terms of high-level intentions, which makes it more understandable by domain experts.

- Q_4 How can process mining techniques be adapted to automate process discovery to support variability and flexibility taking into account the users' intentions and strategies? Process mining enables the design of process from event logs resulting from information systems.
  - $H_{4.1}$ Process mining approaches only consider activity-oriented process models.
  - $H_{4.2}$ Process mining algorithms can be adapted to discover intention-oriented process models.

  A tool, called Map Miner Tool, was designed and developed to automate MMM. It enables applying the proposed approach to obtain the personalized Map process model only from users' traces, by adjusting some parameters.

- Another contribution of this thesis is discovering the most likely sequences of strategies related to the sequence of activities for a given user. In other words, discovering a path in the Map process model comprising a sequence of strategies executed one after the other by a given user. In the future, this will enables to provide a recommender system and assist users in each step of the process enactment.

The entire method is applied on a laboratory context to validate the method and on a real-life traces to demonstrate the scalability of the proposed approach. A qualitative experiment was conducted on the case study to evaluate the perception, effectiveness and usability of MMM in practical use. Some analysis on the discovered process models are also investigated, such as conformance checking and users' behaviors analysis to enhance prescribed models. This show the wide range of usability for the discovered Map and the possibilities of extensions of its performance.

7.2 Open Issues

This thesis focused on the reconstruction of Map process models in an semi-automatic way. In other words, the proposed method finds the relationships between activities to discover the strategies and where they lead, *i.e.* the intentions. However, the names of these strategies and intentions are still inferred manually in the sense that the proposed method is able to extract automatically some topics related to each strategy. This establishes a preliminary base to infer manually the names of strategies and intentions. In the future, this procedure can be fully automated by building sophisticated ontologies from these discovered topics. These ontologies should take into account the context in which the processes are enacted as well as the situation at hand. This will make the discovered Map more context-sensitive.

This thesis mainly focused on the discovery of intentional process models. However, the usefulness of the MMM is not only limited to process discovery. In the future the discovered process models can be useful for multiple concerns. For instance, at the project management level, it allows checking the alignment between prescribed process models and what stakeholders actually do; or at the application level, monitoring users and providing run-time recommendations. This will help improving the software usability by using anterior developers' activities as a guideline by assisting the novice or unfamiliar users. For example, when users' intentions are known, they can be recommended which strategies and activities that might be useful to fulfill their intentions. This guideline is adapted to the users' context taking into account the experiences of previous users and the actual users' intentions. These phases can be automated and integrated as modules of MMM. Furthermore, the discovered Map enables users to be more efficient in their tasks by adapting the system to the users' needs. Assisting users step by step using a Map increases their confidence and satisfaction in the enactment of a process. All these points contribute to improving the usability of the software products.

New applications of intention mining will be found in the near future: mining intentions might also help improving guidance, provide better recommendations, facilitate process modeling, identify the gap between a prescribed business requirements and actual information systems users' goals, help users in a pro-active way, to monitor the intentions of users, and many more.

Chapter 8

Appendix A : Particular Classes

In this Appendix, we present some important snippets, which show how different parts of Map Miner tool are developed : as the snippets for the initialization of the transition and emission matrices. The initialization of these matrices is important as it affects the convergence of BWA. These snippets illustrate partly the construction of pseudo-Map sections by Deep Miner algorithm from BWA results, and Map process model from sections of pseudo-Map.

A snippet of Java code displaying the initialization of transition matrix:

```
System.out.println("************Init transition matrix****************");
        double T_init[][] = new double[Nb_Hs][Nb_Hs];
        for (int i = 0; i < Nb_Hs; i++) {
            for (int j = 0; j < Nb_Hs; j++) {
                if (i == j) {
                    if (i == Nb_Hs - 1) {
                        T_init[i][j] = 1;
                    } else {
                        T_init[i][j] = 0.8;
                    }
                } else {
                    if (i == Nb_Hs - 1) {
                        T_init[i][j] = 0;
                    } else {
                        T_init[i][j] = 0.2 / (Nb_Hs - 1);
                    }
                }
                System.out.print(" " + T_init[i][j] + " ");
            }
            System.out.println();
        }
```

A snippet of Java code displaying the initialization of emission matrix:

```
System.out.println("************Init emission matrix****************");
        // Init emission matrix

        double E_init[][] = new double[Nb_Hs][Nb_Act + 2];
        for (int i = 0; i < Nb_Hs; i++) {
            for (int j = 0; j < Nb_Act + 2; j++) {
                if ((i == 0) || (i == Nb_Hs - 1)) {
                    E_init[i][j] = 0;
                } else {
                    E_init[i][j] = 1.0 / (Nb_Act + 2);
                }
                if (((i == 0) && (j == 0)) || ((i == Nb_Hs - 1) && (j ==
                    Nb_Act + 1))) {
                    E_init[i][j] = 1;
                }
                System.out.print(" " + E_init[i][j] + " ");
            }

            System.out.println();
        }
```

A snippet of Java code displaying the beggining of the construction of the sections with repect to the minimum probability:

```
 System.out.println("**************target**********************");
        System.out.println(target);

        int nbRowSections = 0;
        for (int x = 0; x < Nb_Strat; x++) {
            for (int y = 0; y < Nb_Strat; y++) {
                if (TransReshape.get(x, y) > proba_min) {
                    nbRowSections++;
                }
            }
        }
        SimpleMatrix sections = new SimpleMatrix(nbRowSections, 3);

        int rowIdx = 0;
        for (int i = 0; i < Nb_Strat; i++) {
            for (int j = 0; j < Nb_Strat; j++) {
                if (TransReshape.get(i, j) > proba_min) {
```

```
                double[] sectionsRow = {target.transpose().get(i), j + 1,
                    target.transpose().get(j)};
                for (int k = 0; k < sectionsRow.length; k++) {
                    System.out.print(sectionsRow[k] + " ");
                }
                System.out.println();
                sections.setRow(rowIdx, 0, sectionsRow);
                rowIdx++;
            }
        }
    }
```

A snippet of Java code displaying the construction of the sections (following):

```
System.out.println("**********Sections**********");
        System.out.println(sections);

//Second step of algorithm : refine identical intentions // get all the source intentions
        int sections_before = 0;
        int sections_now = 1;
        double[][] source_intentionDouble = zeros(1, sections.numRows());

        while (sections_before != sections_now) {
            int nb_sections = sections.numRows();
            int cols = sections.numCols();
            for (int i = 0; i < nb_sections; i++) {
                // source_intention [i][i] = sections.get(i,1);
                source_intentionDouble[0][i] = sections.get(i, 0);
            }
```

A snippet of Java code displaying the comparison the sets of exiting sections:

```
if (rowsa == rowsb) {

    int Ba = 0;
      Ba = ((SimpleMatrix) exit_from[i]).numRows();
      SimpleMatrix exit_fromDiff = compareMTX(((SimpleMatrix) exit_from[
          i]), ((SimpleMatrix) exit_from[j]));
```

```
      double exit_fromSum = sum(exit_fromDiff);
      if (exit_fromSum == rowsa * rowsb) {
            SimpleMatrix MTX = compareMTXDouble(getCol(sections, 1),
                source_intention_unique.get(i));
            System.out.println(MTX);
              int[] MTXFind = find(MTX, 1);
                    for (int m = 0; m < MTXFind.length; m++) {
                       if (MTXFind[m] != 0) {
                          sections.set(m, 0, source_intention_unique.get(j)
                              );
                                  }
                           }

    SimpleMatrix MTX2 = compareMTXDouble(getCol(sections, 3),
        source_intention_unique.get(i));
          System.out.println("MTX2");
          System.out.println(MTX2);
          int[] MTX2Find = find(MTX2, 1);
          for (int m = 0; m < MTX2Find.length; m++) {
             if (MTX2Find[m] != 0) {
               sections.set(m, 2, source_intention_unique.get(j)); }
 } } } } } }
```

A snippet of Java code displaying the construction of a pseudo−Map from BWA:

```
 source_intentions = getCol(sections, 1).transpose();
        SimpleMatrix source_intention_unique = unique(
            source_intentions);
        // System.out.println(source_intention_unique.numCols());
        int cpt = 0;
        while (new_number_of_sections < previous_number_of_sections) {
            previous_number_of_sections = new_number_of_sections;

            for (int i = 1; i < source_intention_unique.numCols(); i++) {
                for (int j = 1; j < source_intention_unique.numCols(); j++)
                    {
                    temp_sections = sections.copy();
                    //temp_sections(find(temp_sections(:,1) == source_intentions
                        (i)),1) = source_intentions(j);
                    findRes = find(getCol(temp_sections, 1),
                        source_intention_unique.get(i));
                    for (int k = 0; k < findRes.length; k++) {
```

```
                    if (findRes[k] != 0) {
                        temp_sections.set(findRes[k] - 1, 0,
                            source_intention_unique.get(j));
                    }
                }
                cpt++;
        findRes = find(getCol(temp_sections, 3), source_intention_unique.
            get(i));
    for (int k = 0; k < findRes.length; k++) {
      if (findRes[k] != 0) {
        temp_sections.set(findRes[k] - 1, 2, source_intention_unique.get
            (j));
                    }
                }
 SimpleMatrix Trans_temp = TransMatrixFromSections(temp_sections,
     Trans);
                if (Trans_BWA == Trans_temp) {
       sections = temp_sections.copy();
         new_number_of_sections = length(sections);
  } } } }
```

A snippet of Java code displaying the classification of sub−intentions into groups of intention:

```
 System.out.println("******subIntention******");
        System.out.println(subIntention);

        int numberSections = sections.numRows();
        int cols = sections.numCols();

        int indexOfStartSubIntention[] = null;
        int indexOfEndSubIntention[] = null;
        for (int i = 2; i < numberSections; i++) {
            indexOfStartSubIntention = find(subIntentionIndexExcluded.
                transpose(), sections.get(i, 0));
            System.out.print(indexOfStartSubIntention[0] + " /// ");
            indexOfEndSubIntention = find(subIntentionIndexExcluded.
                transpose(), sections.get(i, 2));
            System.out.println(indexOfEndSubIntention[0]);
            int sum = numberSubIntentions + indexOfEndSubIntention[0];
            subIntention.set(indexOfStartSubIntention[0] - 1, sum - 1, 1);
            subIntention.set(indexOfEndSubIntention[0] - 1,
                indexOfStartSubIntention[0] - 1, 1);
```

```
}

for (int i = 0; i < numberSubIntentions; i++) {
    subIntention.set(i, i, 1);
    subIntention.set(i, numberSubIntentions + i, 1);
}

System.out.println("********subIntention*********");
System.out.println(subIntention);

// %% Classify sub-intentions in groups of intention with K-mean
   algorithm
double[][] subIntentionDoubledd = SimpleMatrix2doubleMD(
    subIntention);
MWNumericArray subIntentionIn = null;
subIntentionIn = new MWNumericArray(subIntentionDoubledd);

Object[] Result = null;
MWNumericArray intentionForSubIntention = null;
kmeansJavaClass x = new kmeansJavaClass();
Result = x.kmeansJava(1, subIntentionIn, numberOfIntentions,
    1000);
intentionForSubIntention = (MWNumericArray) Result[0];

System.out.println("*******intentionForSubIntention********");
System.out.println(intentionForSubIntention);

//%% Generate the new map
for (int i = 0; i < numberSections; i++) {

    indexOfStartSubIntention = find(subIntentionIndexExcluded.
        transpose(), sections.get(i, 0));
    indexOfEndSubIntention = find(subIntentionIndexExcluded.
        transpose(), sections.get(i, 2));
    System.out.println("i = " + i);
    for (int h = 0; h < indexOfStartSubIntention.length; h++) {
        System.out.print("indexOfStartSubIntention = " +
            indexOfStartSubIntention[0] + " ");
    }
    System.out.println();
    for (int h = 0; h < indexOfEndSubIntention.length; h++) {
        System.out.print("indexOfEndSubIntention = " +
            indexOfEndSubIntention[0] + " ");
    }
```

```
        System.out.println();

        if (indexOfStartSubIntention[0] != 0) {
            //sections(i,1) = intentionForSubIntention(
                indexOfStartSubIntention);
            sections.set(i, 0, intentionForSubIntention.getInt(
                indexOfStartSubIntention[0]));
        }
        if (indexOfEndSubIntention[0] != 0) {
            sections.set(i, 2, intentionForSubIntention.getInt(
                indexOfStartSubIntention[0]));
        }

    }
```

Chapter 9

Appendix B : Developers' Questionnaire

This Appendix presents the questions that were asked during the interviews with developers.

- Questions about work habits

  1. Can you briefly describe the activities of your company and yourself?
  2. When you develop software or system what are the steps you pass through?
  3. What are the design methods that you implement in your team?
  4. Do you use design methods of software development? Why are you using them? Why do you not use them?
  5. In which contexts or for which types of projects do you use design methods?
  6. How do you implement the monitoring of your method?
  7. When using a design methodology, do you apply it to the letter? Do you customize it regarding your needs?
  8. Do you present, even partially, your methods to other people?
  9. Do you describe, even partially, your own methods?

- Questions about Map process model

  1. Have you heard about process modeling so far? If so, in which contexts? Did you use it before?
  2. What do you think about Map process model? What might be the benefits of modeling the development processes by Map process model regarding your work habits? What are the disadvantages?
  3. Do you think the use of Map process model requires a learning phase? Is that clear and easy to understand? Is that an intuitive model?
  4. How long takes the appropriation of the learning phase of Map?
  5. Are you ready to describe the operation of Map process model to a third party?

6. Would you wish using Map process model in your development environment? Would you recommend using this process model?

- Questions about the Eclipse Map

1. What is your first impression of the Map?
2. Do you think that the Map reflects the developers' behaviors?
3. Do you think that the Map is useful for a novice developer of Eclipse?
4. Do you think that the Map is helpful to improve the quality of the development method and the best practices of developers?
5. Does the Map help you understanding the development process? Why? For which kind of application would you use it?
6. What is your perception of the concepts represented on the Map?
7. What do you think about the relations between activities and strategies? What about the intentions?
8. What is your perception about the name of strategies and intentions? Are they comprehensive? Do they represent the developers' habits?
9. What do you think about the observations?
10. Do you think that the Map can help you to avoid making mistakes? Why?
11. What do you think of the level of abstraction for the intentions?
12. Do you think the Map is consistent with the usual development method?
13. Is the Map easy to use?
14. Do you need any help to use the Map?
15. Does the Map help you to formalize problems? Which are these problems?

Chapter 10

Appendix C : Journal and Conference Publications

- G. Khodabandelou, C. Hug, C. Salinesi, "A Novel Approach to Process Mining: Intentional Process Models Discovery", long paper, *Proc. of the 8th IEEE International Conference on Research Challenges in Information Systems (RCIS)*, May 2014, Marrakesh, Morocco.
- G. Khodabandelou, C. Hug, R. Deneckère, C. Salinesi, "Supervised vs. Unsupervised Learning for Intentional Process Models Discovery", BPMDS in conjunction with *26th International Conference on Advanced Information Systems Engineering (CAiSE)*, June 2014, Thessaloniki, Creek.
- G. Khodabandelou, C. Hug, R. Deneckère, C. Salinesi, "Unsupervised Discovery of Intentional Process Model from Event Logs", MSR/ICSE (*the 36st International Conference on Software Engineering*), June 2014, Hyderabad, India.
- R. Deneckère, C. Hug, G. Khodabandelou, C. Salinesi, "Intention Mining: Process Model Discovery Using Supervised Learning", *International Journal of Information System Modeling and Design (IJISMD)*, 2014.
- G. Khodabandelou, C. Hug, R. Deneckère, C. Salinesi, "Supervised Intentional Process Models Discovery using Hidden Markov Models", *Proc. of the Seventh IEEE International Conference on Research Challenges in Information Science (RCIS)*, May 2013, Paris, France. **Best Paper Award**
- G. Khodabandelou, C. Hug, R. Deneckère, C. Salinesi, M. Bajec, E. Kornyshova, M. Jankovic, "COTS Products to Trace Method Enactment: Review and Selection", long paper, *Proc. of the 21st European Conference on Information Systems (ECIS)*, June 2013, Utrecht, Netherlands.
- G. Khodabandelou, C. Hug, R. Deneckère, C. Salinesi, "Process Mining versus Intention Mining", *Proc. of Exploring Modelling Methods for Systems Analysis and Design (EMMSAD)*, June 2013, Valencia, Spain.
- G. Khodabandelou, "Contextual Recommendations using Intention Mining on Process Traces", doctoral consortium, *Proc. of the Seventh IEEE International Conference on Research Challenges in Information Science (RCIS)*, May 2013, Paris, France.

- Jankovic, M., Bajec, M., G. Khodabandelou, R. Deneckère, C. Hug, C. Salinesi, "Intelligent Agile Method Framework", *Proc. of the 8st International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, July 2013, Angers, France.

- G. Khodabandelou, C. Hug, R. Deneckère, C. Salinesi, "Découverte Supervisée des Modèles de Processus Intentionnels Basée sur les Modèles de Markov Cachés", *Proc. of the trente-et-unième congrès INFORSID*, May 2013, Paris, France.

Bibliography

[Agrawal 1998] Rakesh Agrawal, Dimitrios Gunopulos and Frank Leymann. Mining process models from workflow logs. Springer, 1998. (Cited on pages 2, 14, 20, 21, 24, 26, 29 and 40.)

[Ajzen 1975] Icek Ajzen and Martin Fishbein. *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research by Martin Fishbein; Icek Ajzen*. 1975. (Cited on page 3.)

[Akkermans 2006] Hans Akkermans and Jaap Gordijn. *Ontology engineering, scientific method and the research agenda*. In Managing Knowledge in a World of Networks, pages 112–125. Springer, 2006. (Cited on page 13.)

[Amyot 2009] Daniel Amyot, Jennifer Horkoff, Daniel Gross and Gunter Mussbacher. *A lightweight GRL profile for i* modeling*. In Advances in Conceptual Modeling-Challenging Perspectives, pages 254–264. Springer, 2009. (Cited on page 45.)

[Arbaoui 1994] Selma Arbaoui and Flavio Oquendo. *Goal oriented vs. activity oriented process modelling and enactment: Issues and perspectives*. In Software Process Technology, pages 171–176. Springer, 1994. (Cited on page 17.)

[Ashkan 2009] Azin Ashkan, Charles LA Clarke, Eugene Agichtein and Qi Guo. *Classifying and characterizing query intent*. In Advances in Information Retrieval, pages 578–586. Springer, 2009. (Cited on page 42.)

[Assar 2000] Saïd Assar, Camille Ben Achour, Samira Si-Said*et al.* *Un Modèle pour la spécification des processus d'analyse des Systèmes d'Information*. In INFORSID, pages 287–301, 2000. (Cited on pages vii, 47 and 60.)

[Baeza-Yates 2005] Ricardo Baeza-Yates. *Applications of web query mining*. In Advances in Information Retrieval, pages 7–22. Springer, 2005. (Cited on page 42.)

[Baeza-Yates 2006] Ricardo Baeza-Yates, Liliana Calderón-Benavides and Cristina González-Caro. *The intention behind web queries*. In String processing and information retrieval, pages 98–109. Springer, 2006. (Cited on pages 7, 32, 34, 36, 37, 38 and 42.)

[Barrios 2004] Judith Barrios and Selmin Nurcan. *Model driven architectures for enterprise information systems*. In Advanced Information Systems Engineering, pages 3–19. Springer, 2004. (Cited on page 45.)

[Bass 2003] Len Bass, Paul Clements and Rick Kazman. *Software Architecture in Practice, 2nd edn. SEI Series in software engineering*, 2003. (Cited on page 125.)

[Baum 1966] Leonard E Baum and Ted Petrie. *Statistical inference for probabilistic functions of finite state Markov chains*. The annals of mathematical statistics, vol. 37, no. 6, pages 1554–1563, 1966. (Cited on page 20.)

[Baum 1970] Leonard E Baum, Ted Petrie, George Soules and Norman Weiss. *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains*. The annals of mathematical statistics, vol. 41, no. 1, pages 164–171, 1970. (Cited on pages 13 and 72.)

[Beck 2001] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries*et al.* *Manifesto for agile software development*. 2001. (Cited on page 15.)

[Bengio 2009] Yoshua Bengio. *Learning deep architectures for AI*. Foundations and trends® in Machine Learning, vol. 2, no. 1, pages 1–127, 2009. (Cited on page 44.)

[Biermann 1972] Alan W Biermann and Jerome A Feldman. *On the synthesis of finite-state machines from samples of their behavior*. Computers, IEEE Transactions on, vol. 100, no. 6, pages 592–597, 1972. (Cited on page 20.)

[Boehm 1988] Barry W. Boehm. *A spiral model of software development and enhancement*. Computer, vol. 21, no. 5, pages 61–72, 1988. (Cited on pages 15 and 17.)

[Bratman 1999] Michael E Bratman. *Intention, plans, and practical reason*. 1999. (Cited on page 32.)

[Bresciani 2004] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia and John Mylopoulos. *Tropos: An agent-oriented software development methodology*. Autonomous Agents and Multi-Agent Systems, vol. 8, no. 3, pages 203–236, 2004. (Cited on page 45.)

[Broder 2002] Andrei Broder. *A taxonomy of web search*. In ACM Sigir forum, volume 36, pages 3–10. ACM, 2002. (Cited on page 34.)

[Burnham 2002] Kenneth P Burnham and David R Anderson. Model selection and multi-model inference: a practical information-theoretic approach. Springer, 2002. (Cited on pages 73 and 75.)

[Cambridge 2013] University Press Cambridge. *Cambridge Advanced Learner's Dictionary and Thesaurus*. url-http://dictionary.cambridge.org/dictionary/british/, December 2013. (Cited on page 11.)

[Carmona 2008] Josep Carmona, Jordi Cortadella and Michael Kishinevsky. *A region-based algorithm for discovering Petri nets from event logs*. In Business Process Management, pages 358–373. Springer, 2008. (Cited on page 22.)

[Chang 2011] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*. ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, no. 3, page 27, 2011. (Cited on page 38.)

[Chen 2002] Zheng Chen, Fan Lin, Huan Liu, Yin Liu, Wei-Ying Ma and Liu Wenyin. *User intention modeling in web applications using data mining*. World Wide Web, vol. 5, no. 3, pages 181–191, 2002. (Cited on pages 7, 32, 35 and 36.)

[Chen 2003] Kevin CW Chen and David YY Yun. *Discovering process models from execution history by graph matching*. In Intelligent Data Engineering and Automated Learning, pages 887–892. Springer, 2003. (Cited on page 41.)

[Chow 1978] Tsun S. Chow. *Testing software design modeled by finite-state machines*. IEEE Trans. Software Eng., vol. 4, no. 3, pages 178–187, 1978. (Cited on pages 11 and 12.)

[Christie 1981] Bruce Christie. *Face to File Communication: A Psychological Approach to Information Systems*. 1981. (Cited on page 3.)

[Chulef 2001] Ada S Chulef, Stephen J Read and David A Walsh. *A hierarchical taxonomy of human goals*. Motivation and Emotion, vol. 25, no. 3, pages 191–232, 2001. (Cited on pages 32 and 49.)

[Clauzel 2009] Damien Clauzel, Karim Sehaba and Yannick Prié. *Modelling and visualising traces for reflexivity in synchronous collaborative systems*. In Intelligent Networking and Collaborative Systems, 2009. INCOS'09. International Conference on, pages 16–23. IEEE, 2009. (Cited on page 10.)

[Conklin 1989] Jeff Conklin and Michael L Begeman. *gIBIS: A tool for all reasons*. Journal of the American Society for Information Science, vol. 40, no. 3, pages 200–213, 1989. (Cited on page 16.)

[Cook 1995] Jonathan E Cook and Alexander L Wolf. *Automating process discovery through event-data analysis*. In Software Engineering, 1995. ICSE 1995. 17th International Conference on, pages 73–73. IEEE, 1995. (Cited on page 20.)

[Cook 1998a] Jonathan E Cook and Alexander L Wolf. *Discovering models of software processes from event-based data*. ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 7, no. 3, pages 215–249, 1998. (Cited on pages 2, 14, 20, 24, 28, 29 and 41.)

[Cook 1998b] Jonathan E Cook and Alexander L Wolf. Event-based detection of concurrency, volume 23. ACM, 1998. (Cited on pages 20, 28, 29 and 30.)

[Cook 1999] Jonathan E Cook and Alexander L Wolf. *Software process validation: quantitatively measuring the correspondence of a process to a model*. ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 8, no. 2, pages 147–176, 1999. (Cited on pages 20 and 27.)

[Cook 2004] Jonathan E Cook, Zhidian Du, Chongbing Liu and Alexander L Wolf. *Discovering models of behavior for concurrent workflows.* Computers in Industry, vol. 53, no. 3, pages 297–319, 2004. (Cited on pages 30 and 40.)

[Curtis 1988] Bill Curtis, Herb Krasner and Neil Iscoe. *A field study of the software design process for large systems.* Communications of the ACM, vol. 31, no. 11, pages 1268–1287, 1988. (Cited on page 17.)

[Curtis 1992] Bill Curtis, Marc I Kellner and Jim Over. *Process modeling.* Communications of the ACM, vol. 35, no. 9, pages 75–90, 1992. (Cited on page 17.)

[Dardenne 1993] Anne Dardenne, Axel Van Lamsweerde and Stephen Fickas. *Goal-directed requirements acquisition.* Science of computer programming, vol. 20, no. 1, pages 3–50, 1993. (Cited on pages 6, 34, 37 and 45.)

[Das 1994a] Sreerupa Das and Michael C Mozer. *A uni ed gradient descent/clustering architecture for finite state machine induction.* Advances in neural information processing systems, vol. 6, pages 19–26, 1994. (Cited on page 20.)

[Das 1994b] Sreerupa Das and Michael C. Mozer. *A Unified Gradient-Descent/Clustering Architecture for Finite State Machine Induction.* In NIPS, pages 19–26. Morgan Kaufmann, 1994. (Cited on page 20.)

[Datta 1998] Anindya Datta. *Automating the Discovery of AS-IS Business Process Models: Probabilistic and Algorithmic Approaches.* Information Systems Research, vol. 9, no. 3, pages 275–301, 1998. (Cited on page 20.)

[Davis 1989] Fred D Davis, Richard P Bagozzi and Paul R Warshaw. *User acceptance of computer technology: a comparison of two theoretical models.* Management science, vol. 35, no. 8, pages 982–1003, 1989. (Cited on pages 3 and 35.)

[de Medeiros 2003] Ana Karla A de Medeiros, Wil MP Van der Aalst and AJMM Weijters. *Workflow mining: Current status and future directions.* In On the move to meaningful internet systems 2003: Coopis, doa, and odbase, pages 389–406. Springer, 2003. (Cited on page 40.)

[de Medeiros 2004] AK Alves de Medeiros, Boudewijn F van Dongen, Wil MP Van der Aalst and AJMM Weijters. *Process mining: Extending the α-algorithm to mine short loops.* Eindhoven University of Technology, Eindhoven, vol. 19, 2004. (Cited on pages 24, 27, 28 and 29.)

[De Medeiros 2005a] AK Alves De Medeiros and Christian W Günther. *Process mining: Using CPN tools to create test logs for mining algorithms.* In Proceedings of the sixth workshop on the practical use of coloured Petri nets and CPN tools (CPN 2005), volume 576, 2005. (Cited on page 31.)

[De Medeiros 2005b] AK Alves De Medeiros and AJMM Weijters. *Genetic process mining*. In Applications and Theory of Petri Nets 2005, volume 3536 of Lecture Notes in Computer Science. Citeseer, 2005. (Cited on pages 22, 24, 26, 29 and 41.)

[de Medeiros 2005c] Ana Karla A de Medeiros, Boudewijn F van Dongen, Wil MP Van der Aalst and AJMM Weijters. *Process mining for ubiquitous mobile systems: an overview and a concrete algorithm*. In Ubiquitous Mobile Information and Collaboration Systems, pages 151–165. Springer, 2005. (Cited on pages 27, 28, 29 and 41.)

[De Medeiros 2006] AK Alves De Medeiros, AJMM Weijters and Wil MP Van der Aalst. *Genetic process mining: a basic approach and its challenges*. In Business Process Management Workshops, pages 203–215. Springer, 2006. (Cited on page 40.)

[De Medeiros 2007] AK Alves De Medeiros, Carlos Pedrinaci, Wil MP Van der Aalst, John Domingue, Minseok Song, Anne Rozinat, Barry Norton and Liliana Cabral. *An outlook on semantic business process mining and monitoring*. In On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, pages 1244–1255. Springer, 2007. (Cited on page 19.)

[Delorenzi 2002] Mauro Delorenzi and Terry Speed. *An HMM model for coiled-coil domains and a comparison with PSSM-based predictions*. Bioinformatics, vol. 18, no. 4, pages 617–625, 2002. (Cited on page 50.)

[Dempster 1977] Arthur P Dempster, Nan M Laird and Donald B Rubin. *Maximum likelihood from incomplete data via the EM algorithm*. Journal of the Royal Statistical Society. Series B (Methodological), pages 1–38, 1977. (Cited on page 72.)

[Deneckère 2010] Rébecca Deneckère and Elena Kornyshova. *Process line configuration: An indicator-based guidance of the intentional model MAP*. In Enterprise, Business-Process and Information Systems Modeling, pages 327–339. Springer, 2010. (Cited on pages 3 and 32.)

[Desel 1995] Jörg Desel. Free-choice petri nets, volume 40. Cambridge university press, 1995. (Cited on pages 22 and 26.)

[Dietz 2005] Jan LG Dietz and Antonia Albani. *Basic notions regarding business processes and supporting information systems*. Requirements Engineering, vol. 10, no. 3, pages 175–183, 2005. (Cited on page 45.)

[Dik 1989] Simon C Dik. The theory of functional grammar. Walter de Gruyter, 1989. (Cited on page 38.)

[Disco 2014] Disco. urlhttp://www.fluxicon.com/disco/, January 2014. (Cited on page 31.)

[Dowson 1987] Mark Dowson. *Iteration in the software process; review of the 3rd International Software Process Workshop.* In Proceedings of the 9th international conference on Software Engineering, pages 36–41. IEEE Computer Society Press, 1987. (Cited on pages 2 and 14.)

[Dustdar 2005] Schahram Dustdar, Thomas Hoffmann and Wil Van der Aalst. *Mining of ad-hoc business processes with TeamLog.* Data & Knowledge Engineering, vol. 55, no. 2, pages 129–158, 2005. (Cited on pages 28, 30 and 40.)

[Eclipse 2013] Eclipse. *Filtered UDC Data.* urlhttp://archive.eclipse.org/projects/usagedata/, 2013. (Cited on pages 7 and 109.)

[Eder 2002] Johann Eder, Georg E Olivotto and Wolfgang Gruber. *A data warehouse for workflow logs.* In Engineering and Deployment of Cooperative Information Systems, pages 1–15. Springer, 2002. (Cited on page 28.)

[Etien 2006] Anne Etien. *Ingénierie de l'alignement: concepts, modèles et processus: la méthode ACEM pour l'alignement d'un système d'information aux processus d'entreprise.* PhD thesis, Paris 1, 2006. (Cited on pages 3 and 17.)

[EUT 2013] EUT. *ProM.* urlhttp://www.processmining.org/prom/start, November 2013. (Cited on page 135.)

[Feather 1987] Martin S Feather. *Language support for the specification and development of composite systems.* ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 9, no. 2, pages 198–234, 1987. (Cited on page 32.)

[Feiler 1993] Peter H Feiler and Watts S Humphrey. *Software process development and enactment: Concepts and definitions.* In Software Process, 1993. Continuous Software Process Improvement, Second International Conference on the, pages 28–40. IEEE, 1993. (Cited on page 14.)

[Fernstrom 1991] Christer Fernstrom and Lennart Ohlsson. *Integration needs in process enacted environments.* In Software Process, 1991. Proceedings. First International Conference on the, pages 142–158. IEEE, 1991. (Cited on page 84.)

[Fickas 1992] Stephen Fickas and B Robert Helm. *Knowledge representation and reasoning in the design of composite systems.* Software Engineering, IEEE Transactions on, vol. 18, no. 6, pages 470–482, 1992. (Cited on page 32.)

[Fillmore 1967] Charles J Fillmore. *The case for case.* 1967. (Cited on page 38.)

[Finkelstein 1991] Anthony Finkelstein, Jeff Kramer and Michael Goedicke. Viewpoint oriented software development. Citeseer, 1991. (Cited on page 15.)

[Fischer 2008] Markus Fischer. *ARIS Process Performance Manager*. In Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008 14th GI/ITG Conference-, pages 1–3. VDE, 2008. (Cited on page 27.)

[Forney Jr 1973] G David Forney Jr. *The viterbi algorithm*. Proceedings of the IEEE, vol. 61, no. 3, pages 268–278, 1973. (Cited on page 87.)

[Fowler 1999] Martin Fowler. Refactoring: improving the design of existing code. Addison-Wesley Professional, 1999. (Cited on page 118.)

[Friedman 1997] Nir Friedman, Dan Geiger and Moises Goldszmidt. *Bayesian network classifiers*. Machine learning, vol. 29, no. 2-3, pages 131–163, 1997. (Cited on pages 11 and 50.)

[Gaaloul 2005] Walid Gaaloul and Claude Godart. *Mining workflow recovery from event based logs*. In Business Process Management, pages 169–185. Springer, 2005. (Cited on pages 24, 29 and 40.)

[Gales 1998] Mark JF Gales. *Maximum likelihood linear transformations for HMM-based speech recognition*. Computer speech & language, vol. 12, no. 2, pages 75–98, 1998. (Cited on page 50.)

[Georgeon 2012] Olivier L Georgeon, Alain Mille, Thierry Bellet, Benoit Mathern and Frank E Ritter. *Supporting activity modelling from activity traces*. Expert Systems, vol. 29, no. 3, pages 261–275, 2012. (Cited on pages 10 and 11.)

[Gilks 1996] Walter R Gilks, Sylvia Richardson and David J Spiegelhalter. Markov chain monte carlo in practice, volume 2. CRC press, 1996. (Cited on page 21.)

[Golani 2003] Mati Golani and Shlomit S Pinter. *Generating a process model from a process audit log*. In Business Process Management, pages 136–151. Springer, 2003. (Cited on pages 28, 29 and 40.)

[González-Caro 2011] Cristina González-Caro and Ricardo Baeza-Yates. *A multi-faceted approach to query intent classification*. In String Processing and Information Retrieval, pages 368–379. Springer, 2011. (Cited on pages 34, 36, 38 and 42.)

[Goutte 2005] Cyril Goutte and Eric Gaussier. *A probabilistic interpretation of precision, recall and F-score, with implication for evaluation*. In Advances in Information Retrieval, pages 345–359. Springer, 2005. (Cited on page 102.)

[Gove 1981] Philip Babcock Gove. Webster's third new international dictionary of the english language, unabridged: A merriam-webster's, volume 1. Merriam-Webster, 1981. (Cited on page 32.)

[Gray 1992] Wayne D Gray, Bonnie E John and Michael E Atwood. *The precis of Project Ernestine or an overview of a validation of GOMS.* In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 307–312. ACM, 1992. (Cited on page 50.)

[Greco 2004] Gianluigi Greco, Antonella Guzzo, Luigi Pontieri and Domenico Sacca. *Mining expressive process models by clustering workflow traces.* In Advances in Knowledge Discovery and Data Mining, pages 52–62. Springer, 2004. (Cited on pages 28, 29 and 41.)

[Greco 2005a] Gianluigi Greco, Antonella Guzzo, Giuseppe Manco and Domenico Sacca. *Mining and reasoning on workflows.* Knowledge and Data Engineering, IEEE Transactions on, vol. 17, no. 4, pages 519–534, 2005. (Cited on page 41.)

[Greco 2005b] Gianluigi Greco, Antonella Guzzo and Luigi Pontieri. *Mining hierarchies of models: From abstract views to concrete specifications.* In Business Process Management, pages 32–47. Springer, 2005. (Cited on page 22.)

[Group 2011] Object Management Group. *usiness Process Model and Notation.* urlhttp://www.omg.org/spec/BPMN/2.0/, December 2011. (Cited on page 39.)

[Group 2013] Object Management Group. *Business Process Model and Notation.* urlhttp://www.omg.org/spec/BPMN/2.0/, December 2013. (Cited on pages 13 and 15.)

[Gruber 1995] Thomas R Gruber. *Toward principles for the design of ontologies used for knowledge sharing?* International journal of human-computer studies, vol. 43, no. 5, pages 907–928, 1995. (Cited on page 12.)

[Günther 2007] Christian W Günther and Wil MP Van der Aalst. *Fuzzy mining–adaptive process simplification based on multi-perspective metrics.* In Business Process Management, pages 328–343. Springer, 2007. (Cited on page 22.)

[Hammori 2004] Markus Hammori, Joachim Herbst and Niko Kleiner. Interactive workflow mining. Springer, 2004. (Cited on pages 28, 30 and 40.)

[Harel 1987] David Harel. *Statecharts: A visual formalism for complex systems.* Science of computer programming, vol. 8, no. 3, pages 231–274, 1987. (Cited on page 15.)

[Hartigan 1979] John A Hartigan and Manchek A Wong. *Algorithm AS 136: A k-means clustering algorithm.* Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, pages 100–108, 1979. (Cited on pages 54 and 86.)

[Hashemi 2008] Ray Hashemi, Azita Bahrami, James LaPlant and Kenneth Thurber. *Discovery of Intent through the Analysis of Visited Sites.* In Hamid R. Arabnia and Ray R. Hashemi, editeurs, IKE, pages 417–422. CSREA Press, 2008. (Cited on pages 7, 32, 34, 35, 36 and 37.)

[Hassine 2002] I Hassine, D Rieu, F Bounaas and O Seghrouchni. *Symphony: a conceptual model based on business components.* In Systems, Man and Cybernetics, 2002 IEEE International Conference on, volume 3, pages 6–pp. IEEE, 2002. (Cited on page 15.)

[Hayashi 2003] Miwa Hayashi. *Hidden markov models to identify pilot instrument scanning and attention patterns.* In Systems, Man and Cybernetics, 2003. IEEE International Conference on, volume 3, pages 2889–2896. IEEE, 2003. (Cited on pages 50 and 66.)

[Henderson-Sellers 1990] Brian Henderson-Sellers and Julian M Edwards. *The object-oriented systems life cycle.* Communications of the ACM, vol. 33, no. 9, pages 142–159, 1990. (Cited on page 15.)

[Herbst 1998] Joachim Herbst and Dimitris Karagiannis. *Integrating machine learning and workflow management to support acquisition and adaptation of workflow models.* In Database and Expert Systems Applications, 1998. Proceedings. Ninth International Workshop on, pages 745–752. IEEE, 1998. (Cited on pages 21, 22, 40, 41 and 51.)

[Herbst 1999] Joachim Herbst and D Karagiannis. *An inductive approach to the acquisition and adaptation of workflow models.* In Proceedings of the IJCAI, volume 99, pages 52–57. Citeseer, 1999. (Cited on pages 21 and 22.)

[Herbst 2000a] Joachim Herbst. *Dealing with concurrency in workflow induction.* In European Concurrent Engineering Conference. SCS Europe. Citeseer, 2000. (Cited on pages 21 and 28.)

[Herbst 2000b] Joachim Herbst. *A machine learning approach to workflow management.* In Machine Learning: ECML 2000, pages 183–194. Springer, 2000. (Cited on pages 20, 21 and 22.)

[Herbst 2004a] Joachim Herbst. Ein induktiver ansatz zur akquisition und adaption von workflow-modellen. Tenea Verlag Ltd., 2004. (Cited on page 21.)

[Herbst 2004b] Joachim Herbst and Dimitris Karagiannis. *Workflow mining with InWoLvE.* Computers in Industry, vol. 53, no. 3, pages 245–264, 2004. (Cited on pages 26, 27, 28, 30, 31 and 41.)

[Hoey 2007] Jesse Hoey and James J Little. *Value-directed human behavior analysis from video using partially observable markov decision processes.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 29, no. 7, pages 1118–1132, 2007. (Cited on page 50.)

[Huff 1987] Karen Huff and Victor R Lesser. *The GRAPPLE plan formalism*. 1987. (Cited on page 16.)

[Hug 2009] Charlotte Hug. *Méthode, modèles et outil pour la méta-modélisation des processus d'ingénierie de systèmes d'information*. PhD thesis, Université Joseph-Fourier-Grenoble I, 2009. (Cited on page 119.)

[Humphrey 1989] Watts S Humphrey and Marc I Kellner. *Software process modeling: principles of entity process models*. In Proceedings of the 11th international conference on Software engineering, pages 331–342. ACM, 1989. (Cited on page 15.)

[Hwang 2004] San-Yih Hwang, Chih-Ping Wei and Wan-Shiou Yang. *Discovery of temporal patterns from process instances*. Computers in Industry, vol. 53, no. 3, pages 345–364, 2004. (Cited on pages 30 and 41.)

[IBM 1999] IBM. *Getting Started With Buildtime*. IBM MQSeries Workflow, 1999. (Cited on page 21.)

[IBM 2014] IBM. urlhttp://publib.boulder.ibm.com/, January 2014. (Cited on page 135.)

[Inglis 2011] Matthew Inglis. *Proof in mathematics education: research, learning and teaching*. Research in Mathematics Education, vol. 13, no. 3, pages 316–320, 2011. (Cited on page 78.)

[Jackson 1995] Michael Jackson. Software requirements and specifications: A lexicon of practice, principles and prejudices, volume 1. ACM Press, 1995. (Cited on page 32.)

[Janković 2013] Marko Janković, Marko Bajec, Ghazaleh Khodabandelou, Rebecca Deneckere, Charlotte Hug, Camille Salinesi*et al.* *Intelligent Agile Method Framework*. In Proceedings of 8th International Conference on Evaluation of Novel Approaches to Software Engineering, pages 1–6, 2013. (Cited on page 3.)

[Jansen 2007] Bernard J Jansen, Danielle L Booth and Amanda Spink. *Determining the user intent of web search engine queries*. In Proceedings of the 16th international conference on World Wide Web, pages 1149–1150. ACM, 2007. (Cited on page 42.)

[Jarke 1992] Matthias Jarke, John Mylopoulos, Joachim W. Schmidt and Yannis Vassiliou. *DAIDA: An environment for evolving information systems*. ACM Transactions on Information Systems (TOIS), vol. 10, no. 1, pages 1–50, 1992. (Cited on pages 2 and 16.)

[Jarke 1993] Matthias Jarke and Klaus Pohl. *Establishing visions in context: towards a model of requirements processes*. In ICIS, pages 23–34, 1993. (Cited on page 3.)

[Jensen 1996a] Finn V Jensen. An introduction to bayesian networks, volume 210. UCL press London, 1996. (Cited on page 11.)

[Jensen 1996b] Kurt Jensen. Coloured petri nets: basic concepts, analysis methods and practical use, volume 1. Springer, 1996. (Cited on page 31.)

[Jensen 2007] Kurt Jensen, Lars Michael Kristensen and Lisa Wells. *Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems.* International Journal on Software Tools for Technology Transfer, vol. 9, no. 3-4, pages 213–254, 2007. (Cited on page 31.)

[Jethava 2011] Vinay Jethava, Liliana Calderón-Benavides, Ricardo A Baeza-Yates, Chiranjib Bhattacharyya and Devdatt P Dubhashi. *Scalable multi-dimensional user intent identification using tree structured distributions.* In SIGIR, pages 395–404, 2011. (Cited on pages 34, 36 and 42.)

[Juang 1991] Biing Hwang Juang and Laurence R Rabiner. *Hidden Markov models for speech recognition.* Technometrics, vol. 33, no. 3, pages 251–272, 1991. (Cited on pages 11, 12 and 50.)

[Kaabi 2007] Rim Samia Kaabi, Carine Souveyet*et al.* *Capturing intentional services with business process maps.* In 1rst IEEE International Conference on Research Challenges in Information Science, 2007. (Cited on page 37.)

[Kathuria 2010] Ashish Kathuria, Bernard J Jansen, Carolyn Hafernik and Amanda Spink. *Classifying the user intent of web queries using¡ IT¿ k¡/IT¿-means clustering.* Internet Research, vol. 20, no. 5, pages 563–581, 2010. (Cited on page 42.)

[Kelley 2008] Richard Kelley, Monica Nicolescu, Alireza Tavakkoli, C King and G Bebis. *Understanding human intentions via hidden markov models in autonomous mobile robots.* In Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on, pages 367–374. IEEE, 2008. (Cited on page 35.)

[Khodabandelou 2013] Ghazaleh Khodabandelou, Charlotte Hug, Rebecca Deneckere, Camille Salinesi*et al.* *Supervised Intentional Process Models Discovery using Hidden Markov Models.* In Proceedings of Seventh International Conference on Research Challenges in Information Science, 2013. (Cited on page 71.)

[Kil 1996] D. H. Kil and F. B. Shin. *Pattern Recognition and Prediction with Applications to Signal Characterization.* Modern Acoustics and Signal Processing, vol. XVI, page 418, 1996. (Cited on page 50.)

[Kornyshova 2007] Elena Kornyshova, Rébecca Deneckère and Camille Salinesi. *Method Chunks Selection by Multicriteria Techniques: an Extension of the*

Assembly-based Approach. In Situational Method Engineering: Fundamentals and Experiences, pages 64–78. Springer, 2007. (Cited on page 45.)

[Kröll 2009] Mark Kröll and Markus Strohmaier. *Analyzing human intentions in natural language text*. In Proceedings of the fifth international conference on Knowledge capture, pages 197–198. ACM, 2009. (Cited on pages 36 and 42.)

[Kruchten 2004] Philippe Kruchten. The rational unified process: an introduction. Addison-Wesley Professional, 2004. (Cited on page 15.)

[Kumar 2006] Nanda Kumar and Izak Benbasat. *Research note: the influence of recommendations and consumer reviews on evaluations of websites*. Information Systems Research, vol. 17, no. 4, pages 425–439, 2006. (Cited on pages 36 and 42.)

[Kunz 1970] Werner Kunz and Horst WJ Rittel. Issues as elements of information systems, volume 131. Institute of Urban and Regional Development, University of California Berkeley, California, 1970. (Cited on page 16.)

[Laflaquière 2006] Julien Laflaquière, Lotfi S Settouti, Yannick Prié and Alain Mille. *Trace-based framework for experience management and engineering*. In Knowledge-Based Intelligent Information and Engineering Systems, pages 1171–1178. Springer, 2006. (Cited on page 11.)

[Lee 1991] Jintae Lee. *Extending the Potts and Bruns model for recording design rationale*. In Software Engineering, 1991. Proceedings., 13th International Conference on, pages 114–125. IEEE, 1991. (Cited on page 33.)

[Lee 2012] Sangkeun Lee. *A generic graph-based multidimensional recommendation framework and its implementations*. In Proceedings of the 21st international conference companion on World Wide Web, pages 161–166. ACM, 2012. (Cited on pages 36 and 42.)

[Li 1999] Cen Li and Gautam Biswas. *Finding behavior patterns from temporal data using hidden markov model based unsupervised classification*. Proceedings of the 1999 CIMA: Computational Intelligence Methods and Applications (Rochester, NY, June 22-25), pages 266–272, 1999. (Cited on page 50.)

[Liu 2004] Hugo Liu and Push Singh. *ConceptNet—a practical commonsense reasoning tool-kit*. BT technology journal, vol. 22, no. 4, pages 211–226, 2004. (Cited on page 36.)

[MacDermid 1984] John A MacDermid and Knut Ripken. Life cycle support in the ada environment. CUP Archive, 1984. (Cited on page 15.)

[Malcolm 1967] Norman Malcolm. *Explaining behavior*. The Philosophical Review, pages 97–104, 1967. (Cited on page 3.)

[Mannila 1997] Heikki Mannila, Hannu Toivonen and A Inkeri Verkamo. *Discovery of frequent episodes in event sequences.* Data Mining and Knowledge Discovery, vol. 1, no. 3, pages 259–289, 1997. (Cited on page 21.)

[Mannila 2001] Heikki Mannila and Dmitry Rusakov. *Decomposition of event sequences into independent components.* In Proceedings of the 1st SIAM ICDM, Chicago, IL. Citeseer, 2001. (Cited on pages 21 and 41.)

[Martelli 2002] Pier Luigi Martelli, Piero Fariselli, Anders Krogh and Rita Casadio. *A sequence-profile-based HMM for predicting and discriminating β barrel membrane proteins.* Bioinformatics, vol. 18, no. suppl 1, pages S46–S53, 2002. (Cited on page 50.)

[Martin 1967] James John Martin. *Bayesian decision problems and Markov chains.* 1967. (Cited on pages 11 and 12.)

[Martin 1991] James Martin. Rapid application development. Macmillan Publishing Company, 1991. (Cited on page 15.)

[Maruster 2001] Laura Maruster, WMP Van der Aalst, AJMM Weijters, Antal van den Bosch and Walter Daelemans. *Automated discovery of workflow models from hospital data.* In Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001), pages 183–190, 2001. (Cited on pages 24 and 40.)

[Maruster 2002] Laura Maruster, AJMM Ton Weijters, WMP Wil Van der Aalst and Antal van den Bosch. *Process mining: Discovering direct successors in process logs.* In Discovery Science, pages 364–373. Springer, 2002. (Cited on page 40.)

[Medeiros 2005] AK Alves Medeiros, Antonius Jozef Martha Maria Weijters and Willibrordus Martinus Pancratius Aalst. Using genetic algorithms to mine process models: representation, operators and results. Beta, Research School for Operations Management and Logistics, 2005. (Cited on pages 27, 29 and 41.)

[Mei 2005] Tao Mei, Xian-Sheng Hua and He-Qin Zhou. *Tracking users' capture intention: a novel complementary view for home video content analysis.* In Proceedings of the 13th annual ACM international conference on Multimedia, pages 531–534. ACM, 2005. (Cited on page 35.)

[Miller 1995] George A Miller. *WordNet: a lexical database for English.* Communications of the ACM, vol. 38, no. 11, pages 39–41, 1995. (Cited on page 36.)

[Mirbel 2006] Isabelle Mirbel and Jolita Ralyté. *Situational method engineering: combining assembly-based and roadmap-driven approaches.* Requirements Engineering, vol. 11, no. 1, pages 58–78, 2006. (Cited on pages 4 and 32.)

[Mobasher 2000] Bamshad Mobasher, Robert Cooley and Jaideep Srivastava. *Automatic personalization based on Web usage mining*. Communications of the ACM, vol. 43, no. 8, pages 142–151, 2000. (Cited on page 20.)

[Mostow 1985] Jack Mostow. *Toward better models of the design process*. AI magazine, vol. 6, no. 1, page 44, 1985. (Cited on page 33.)

[Mulyar 2008] Nataliya Mulyar, Maja Pesic, Wil MP Van der Aalst and Mor Peleg. *Declarative and procedural approaches for modelling clinical guidelines: addressing flexibility issues*. In Business Process Management Workshops, pages 335–346. Springer, 2008. (Cited on page 19.)

[Murphy-Hill 2008] Emerson Murphy-Hill and Andrew Black. *Breaking the barriers to successful refactoring*. In Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on, pages 421–430. IEEE, 2008. (Cited on page 118.)

[Murphy 2002] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, 2002. (Cited on pages 11 and 66.)

[Mylopoulos 1992] John Mylopoulos, Lawrence Chung and Brian Nixon. *Representing and using nonfunctional requirements: A process-oriented approach*. Software Engineering, IEEE Transactions on, vol. 18, no. 6, pages 483–497, 1992. (Cited on page 33.)

[Myung 2003] In Jae Myung. *Tutorial on maximum likelihood estimation*. Journal of Mathematical Psychology, vol. 47, no. 1, pages 90–100, 2003. (Cited on page 71.)

[Najar 2011] Salma Najar, Manuele Kirsch-Pinheiro and Carine Souveyet. *Towards semantic modeling of intentional pervasive information systems*. In Proceedings of the 6th International Workshop on Enhanced Web Service Technologies, pages 30–34. ACM, 2011. (Cited on pages 3 and 32.)

[Nurcan 2005] Selmin Nurcan, Anne Etien, Rim Kaabi, Iyad Zoukar and Colette Rolland. *A strategy driven business process modelling approach*. Business Process Management Journal, vol. 11, no. 6, pages 628–649, 2005. (Cited on page 37.)

[Object Management Group 2013] Inc. Object Management Group. *BMPN*. urlhttp://www.bpmn.org, December 2013. (Cited on page 22.)

[Olle 1988] T William Olle and T William Olle. Information systems methodologies: a framework for understanding, volume 8. Addison-Wesley Reading, 1988. (Cited on page 14.)

[Outmazgin 2013] Nesi Outmazgin and Pnina Soffer. *Business Process Workarounds: What Can and Cannot Be Detected by Process Mining*. In Enterprise, Business-Process and Information Systems Modeling, pages 48–62. Springer, 2013. (Cited on pages 7, 32, 33 and 42.)

[Park 2010] Kinam Park, Taemin Lee, Soonyoung Jung, Heuiseok Lim and Sangyep Nam. *Extracting Search Intentions from Web Search Logs*. In Information Technology Convergence and Services (ITCS), 2010 2nd International Conference on, pages 1–6. IEEE, 2010. (Cited on pages 34, 35, 36, 37 and 42.)

[Pesic 2006] Maja Pesic and Wil MP Van der Aalst. *A declarative approach for flexible business processes management*. In Business Process Management Workshops, pages 169–180. Springer, 2006. (Cited on pages 22 and 23.)

[Peterson 1981] James L Peterson. *Petri net theory and the modeling of systems*. 1981. (Cited on pages 22 and 23.)

[Plihon 1996] V Plihon. *Un environnement pour l'ingénierie des méthodes*. PhD thesis, Université Paris 1 Panthéon-Sorbonne, 1996. (Cited on pages 3, 16 and 18.)

[Pohl 1999] Klaus Pohl, Klaus Weidenhaupt, Ralf Dömges, Peter Haumer, Matthias Jarke and Ralf Klamma. *PRIME—toward process-integrated modeling environments: 1*. ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 8, no. 4, pages 343–410, 1999. (Cited on page 16.)

[Potts 1988] Colin Potts and Glenn Bruns. *Recording the reasons for design decisions*. In Proceedings of the 10th international conference on Software engineering, pages 418–427. IEEE Computer Society Press, 1988. (Cited on page 16.)

[Prakash 2006] Naveen Prakash and Colette Rolland. *Systems Design for Requirements Expressed as a Map*. In Emerging Trends and Challenges in Information Technology Management, page 3, 2006. (Cited on page 45.)

[Rabiner 1986] Lawrence Rabiner and B Juang. *An introduction to hidden Markov models*. ASSP Magazine, IEEE, vol. 3, no. 1, pages 4–16, 1986. (Cited on page 73.)

[Rabiner 1989] Lawrence R Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition*. Proceedings of the IEEE, vol. 77, no. 2, pages 257–286, 1989. (Cited on pages 7, 21, 50, 52 and 74.)

[Ralph 2008] Paul Ralph and Yair Wand. *A teleological process theory of software development*. 2008. (Cited on page 2.)

[Ralyté 1999] Jolita Ralyté. *Reusing scenario based approaches in requirement engineering methods: CREWS method base*. In Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on, pages 305–309. IEEE, 1999. (Cited on pages 3 and 16.)

[Ralyté 2003] Jolita Ralyté, Rébecca Deneckère and Colette Rolland. *Towards a generic model for situational method engineering*. In Advanced Information Systems Engineering, pages 95–110. Springer, 2003. (Cited on pages 3 and 32.)

[Rebstock 2008] Michael Rebstock, Janina Fengel and Heiko Paulheim. Ontologies-based business integration. Springer, 2008. (Cited on page 13.)

[Rissanen 1978] Jorma Rissanen. *Modeling by shortest data description*. Automatica, vol. 14, no. 5, pages 465–471, 1978. (Cited on page 27.)

[Rolland 1993] Colette Rolland. *Modeling the requirements engineering process*. In Information Modelling and Knowledge Bases V: Principles and Formal Techniques: Results of the 3rd European-Japanese Seminar, Budapest, Hungary, May, pages 85–96, 1993. (Cited on pages 3, 6 and 32.)

[Rolland 1994] Colette Rolland and N Prakash. *A contextual approach for the requirements engineering process*. In SEKE, pages 28–35. Citeseer, 1994. (Cited on pages 14 and 16.)

[Rolland 1998a] Colette Rolland. *A comprehensive view of process engineering*. In Advanced Information Systems Engineering, pages 1–24. Springer, 1998. (Cited on pages 14, 16, 17, 18, 49 and 83.)

[Rolland 1998b] Colette Rolland, C Ben Achour, Corine Cauvet, Jolita Ralyté, Alistair Sutcliffe, Neil Maiden, Matthias Jarke, Peter Haumer, Klaus Pohl, Eric Dubois*et al.* *A proposal for a scenario classification framework*. Requirements Engineering, vol. 3, no. 1, pages 23–47, 1998. (Cited on page 4.)

[Rolland 1999] Colette Rolland, Naveen Prakash and Adolphe Benjamen. *A multi-model view of process modelling*. Requirements Engineering, vol. 4, no. 4, pages 169–187, 1999. (Cited on pages 3, 6, 7, 14, 16, 37, 38, 45 and 51.)

[Rolland 2000] Colette Rolland, Selmin Nurcan and Georges Grosz. *A decision-making pattern for guiding the enterprise knowledge development process*. Information and software technology, vol. 42, no. 5, pages 313–331, 2000. (Cited on page 16.)

[Rolland 2005a] Colette Rolland. *L'ingénierie des méthodes: une visite guidée*. e-TI, vol. 1, 2005. (Cited on pages 2, 14, 15 and 16.)

[Rolland 2005b] Colette Rolland and Camille Salinesi. *Modeling goals and reasoning with them*. In Engineering and Managing Software Requirements, pages 189–217. Springer, 2005. (Cited on pages 3, 4, 6, 7, 33, 48 and 49.)

[Rolland 2007] Colette Rolland. *Capturing system intentionality with maps.* In Conceptual modelling in Information Systems engineering, pages 141–158. Springer, 2007. (Cited on pages vii, 3, 16, 33, 37, 38, 45, 46, 47 and 48.)

[Rolland 2009] Colette Rolland and Camille Salinesi. *Supporting requirements elicitation through goal/scenario coupling.* In Conceptual Modeling: Foundations and Applications, pages 398–416. Springer, 2009. (Cited on page 7.)

[Rolland 2010] Colette Rolland, Manuele Kirsch-Pinheiro and Carine Souveyet. *An intentional approach to service engineering.* Services Computing, IEEE Transactions on, vol. 3, no. 4, pages 292–305, 2010. (Cited on pages 3 and 32.)

[Roques 2004] Pascal Roques and Franck Vallée. *UML 2 en action.* De l'analyse des besoins í*** la conception J2EE, 3ème édition Eyrolles, 2004. (Cited on page 15.)

[Rose 1991] Thomas Rose, Matthias Jarke, Michael Gocek, Carlos Maltzahn and Hans W Nissen. *A decision based configuration process environment.* Software Engineering Journal, vol. 6, no. 5, pages 332–346, 1991. (Cited on page 16.)

[Rousseau 2001] Denise M Rousseau. *Schema, promise and mutuality: The building blocks of the psychological contract.* Journal of occupational and organizational psychology, vol. 74, no. 4, pages 511–541, 2001. (Cited on page 35.)

[Royce 1970] Winston W Royce. *Managing the development of large software systems.* In proceedings of IEEE WESCON, volume 26. Los Angeles, 1970. (Cited on page 15.)

[Rozinat 2007] Anne Rozinat, AK Alves de Medeiros, Christian W Günther, AJMM Weijters and Wil MP Van der Aalst. Towards an evaluation framework for process mining algorithms. Beta, Research School for Operations Management and Logistics, 2007. (Cited on pages 2 and 66.)

[Rozinat 2008a] Anne Rozinat, RS Mans, Minseok Song and Wil MP Van der Aalst. *Discovering colored Petri nets from event logs.* International Journal on Software Tools for Technology Transfer, vol. 10, no. 1, pages 57–74, 2008. (Cited on page 31.)

[Rozinat 2008b] Anne Rozinat, M Veloso and Wil MP Van der Aalst. *Evaluating the quality of discovered process models.* In 2nd Intl. Workshop on the Induction of Process Models, Antwerp, Belgium, pages 45–52. Citeseer, 2008. (Cited on page 52.)

[Rozinat 2010] Anne Rozinat. *Process Mining Conformance and Extension.* PhD thesis, Technische Universiteit Eindhoven, 2010. (Cited on pages 14, 21, 50 and 52.)

[Sadikov 2010] Eldar Sadikov, Jayant Madhavan, Lu Wang and Alon Halevy. *Clustering query refinements by user intent*. In Proceedings of the 19th international conference on World wide web, pages 841–850. ACM, 2010. (Cited on pages 36 and 42.)

[Salinesi 2003] Camille Salinesi and Colette Rolland. *Fitting business models to system functionality exploring the fitness relationship*. In Advanced Information Systems Engineering, pages 647–664. Springer, 2003. (Cited on pages 3 and 32.)

[Scheer 2014] I.D.S. Scheer. urlhttp://www.ids-scheer.com, January 2014. (Cited on page 31.)

[Schein 2001] Andrew I Schein, Alexandrin Popescul and Lyle H Ungar. *PennAspect: Two-way aspect model implementation*. 2001. (Cited on page 38.)

[Schimm 2003] Guido Schimm. *Mining most specific workflow models from event-based data*. In Business process management, pages 25–40. Springer, 2003. (Cited on pages 22, 27, 30 and 41.)

[Schimm 2004] Guido Schimm. *Mining exact models of concurrent workflows*. Computers in Industry, vol. 53, no. 3, pages 265–281, 2004. (Cited on pages 22, 28, 30 and 40.)

[Schonenberg 2008] Helen Schonenberg, Barbara Weber, Boudewijn van Dongen and Wil Van der Aalst. *Supporting flexible processes through recommendations based on history*. In Business Process Management, pages 51–66. Springer, 2008. (Cited on pages 19 and 20.)

[Schwaber 2002] Ken Schwaber and Mike Beedle. Agile software development with scrum, volume 1. Prentice Hall Upper Saddle River, 2002. (Cited on page 15.)

[Shen 2011] Yan Shen, Yuefeng Li, Yue Xu, Renato Iannella, Abdulmohsen Algarni and Xiaohui Tao. *An ontology-based mining approach for user search intent discovery*. In ADCS 2011: Proceedings of the Sixteenth Australasian Document Computing Symposium, pages 39–46, 2011. (Cited on page 42.)

[Soffer 2005] Pnina Soffer and Colette Rolland. *Combining intention-oriented and state-based process modeling*. In Conceptual Modeling–ER 2005, pages 47–62. Springer, 2005. (Cited on pages 19 and 44.)

[Song 2008] Minseok Song and Wil MP Van der Aalst. *Towards comprehensive support for organizational mining*. Decision Support Systems, vol. 46, no. 1, pages 300–317, 2008. (Cited on page 20.)

[Song 2009] Minseok Song, Christian W Günther and Wil MP Van der Aalst. *Trace clustering in process mining*. In Business Process Management Workshops, pages 109–120. Springer, 2009. (Cited on page 22.)

[Souveyet 2006] Carine Souveyet. *Contributions à l'amélioration de l'ingénierie des SI*. PhD thesis, Habilitation à diriger les Recherches, Université Paris 1 Panthéon-Sorbonne, 2006. (Cited on page 18.)

[Staffware 2014] Staffware. url http://www.staffware.com, January 2014. (Cited on page 31.)

[Strohmaier 2009] Markus Strohmaier and Mark Kröll. *Studying databases of intentions: do search query logs capture knowledge about common human goals?* In Proceedings of the fifth international conference on Knowledge capture, pages 89–96. ACM, 2009. (Cited on page 42.)

[Strohmaier 2012] Markus Strohmaier and Mark Kröll. *Acquiring knowledge about human goals from search query logs*. Information Processing & Management, vol. 48, no. 1, pages 63–82, 2012. (Cited on pages 34, 36, 37, 39 and 42.)

[Swanson 1974] E Burton Swanson. *Management information systems: appreciation and involvement*. Management Science, vol. 21, no. 2, pages 178–188, 1974. (Cited on pages 3 and 35.)

[Taylor 1964] Charles Taylor. The explanation of behaviour. Humanities Press, 1964. (Cited on page 2.)

[Thevenet 2007a] Laure-Hélène Thevenet, Ines Gam, Camille Salinesi*et al. Strategic Alignment Documentation.* In RCIS, pages 331–342, 2007. (Cited on page 33.)

[Thevenet 2007b] Laure-Hélène Thevenet and Camille Salinesi. *Aligning IS to organization's strategy: the INSTAL method*. In Advanced Information Systems Engineering, pages 203–217. Springer, 2007. (Cited on page 3.)

[Tibco 2000] Tibco. *TIB/InConcert Process Designer Users' Guide*. 2000. (Cited on page 21.)

[Tiwari 2008] A Tiwari, CJ Turner and B Majeed. *A review of business process mining: state-of-the-art and future trends*. Business Process Management Journal, vol. 14, no. 1, pages 5–22, 2008. (Cited on pages 14 and 21.)

[Tsai 2006] Anni Tsai, Jiacun Wang, William Tepfenhart and Daniela Rosea. *EPC Workflow model to WIFA model conversion*. In Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on, volume 4, pages 2758–2763. IEEE, 2006. (Cited on page 23.)

[Tversky 1974] Amos Tversky and Daniel Kahneman. *Judgment under uncertainty: Heuristics and biases*. science, vol. 185, no. 4157, pages 1124–1131, 1974. (Cited on page 71.)

[UDC 2013] Eclipse UDC. *Filtered UDC Data*. url-http://www.eclipse.org/org/usagedata/, 2013. (Cited on page 109.)

[Vakilian 2012] Mohsen Vakilian, Nicholas Chen, Stas Negara, Balaji Ambresh Rajkumar, Brian P Bailey and Ralph E Johnson. *Use, disuse, and misuse of automated refactorings*. In Software Engineering (ICSE), 2012 34th International Conference on, pages 233–243. IEEE, 2012. (Cited on page 118.)

[Van der Aalst 1999] Wil MP Van der Aalst. *Formalization and verification of event-driven process chains*. Information and Software technology, vol. 41, no. 10, pages 639–650, 1999. (Cited on page 22.)

[Van der Aalst 2001] Wil MP Van der Aalst and Twan Basten. *Identifying commonalities and differences in object life cycles using behavioral inheritance*. In Applications and Theory of Petri Nets 2001, pages 32–52. Springer, 2001. (Cited on page 27.)

[Van der Aalst 2002a] Wil MP Van der Aalst and Boudewijn F van Dongen. *Discovering workflow performance models from timed logs*. In Engineering and Deployment of Cooperative Information Systems, pages 45–63. Springer, 2002. (Cited on page 22.)

[Van der Aalst 2002b] Wil MP Van der Aalst, AJMM Weijters and Laura Maruster. *Workflow mining: Which processes can be rediscovered*. Rapport technique, Citeseer, 2002. (Cited on pages 22, 26, 29 and 41.)

[Van der Aalst 2003] Wil MP Van der Aalst, Boudewijn F van Dongen, Joachim Herbst, Laura Maruster, Guido Schimm and AJMM Weijters. *Workflow mining: a survey of issues and approaches*. Data & knowledge engineering, vol. 47, no. 2, pages 237–267, 2003. (Cited on page 20.)

[Van der Aalst 2004a] Wil Van der Aalst, Ton Weijters and Laura Maruster. *Workflow mining: Discovering process models from event logs*. Knowledge and Data Engineering, IEEE Transactions on, vol. 16, no. 9, pages 1128–1142, 2004. (Cited on pages 2, 20, 21 and 40.)

[Van der Aalst 2004b] Wil MP Van der Aalst and Minseok Song. *Mining Social Networks: Uncovering interaction patterns in business processes*. In Business Process Management, pages 244–260. Springer, 2004. (Cited on pages 27, 30 and 40.)

[Van der Aalst 2004c] Wil MP Van der Aalst and AJMM Weijters. *Process mining: a research agenda*. Computers in industry, vol. 53, no. 3, pages 231–244, 2004. (Cited on pages vii, 2, 14, 23, 24, 25, 26 and 27.)

[Van der Aalst 2005a] Wil MP Van der Aalst. *Business alignment: using process mining as a tool for Delta analysis and conformance testing*. Requirements Engineering, vol. 10, no. 3, pages 198–211, 2005. (Cited on pages 19, 27, 29 and 40.)

[Van der Aalst 2005b] Wil MP Van der Aalst, AK Alves de Medeiros and AJMM Weijters. *Genetic process mining.* In Applications and Theory of Petri Nets 2005, pages 48–69. Springer, 2005. (Cited on pages 24, 26, 27, 28 and 30.)

[Van der Aalst 2005c] Wil MP Van der Aalst and Ana Karla A de Medeiros. *Process mining and security: Detecting anomalous process executions and checking process conformance.* Electronic Notes in Theoretical Computer Science, vol. 121, pages 3–21, 2005. (Cited on pages 24, 26, 27, 30 and 41.)

[Van der Aalst 2005d] Wil MP Van der Aalst, Hajo A Reijers and Minseok Song. *Discovering social networks from event logs.* Computer Supported Cooperative Work (CSCW), vol. 14, no. 6, pages 549–593, 2005. (Cited on page 20.)

[Van der Aalst 2009] Wil MP Van der Aalst, Boudewijn F van Dongen, Christian W Günther, Anne Rozinat, Eric Verbeek and Ton Weijters. *ProM: The Process Mining Toolkit.* BPM (Demos), vol. 489, 2009. (Cited on page 31.)

[Van der Aalst 2010] Wil MP Van der Aalst, Vladimir Rubin, HMW Verbeek, Boudewijn F van Dongen, Ekkart Kindler and Christian W Günther. *Process mining: a two-step approach to balance between underfitting and overfitting.* Software & Systems Modeling, vol. 9, no. 1, pages 87–111, 2010. (Cited on page 22.)

[Van der Aalst 2011a] Wil Van der Aalst and Christian Stahl. Modeling business processes: a petri net-oriented approach. The MIT Press, 2011. (Cited on pages 19 and 23.)

[Van der Aalst 2011b] Wil MP Van der Aalst, MH Schonenberg and Minseok Song. *Time prediction based on process mining.* Information Systems, vol. 36, no. 2, pages 450–475, 2011. (Cited on page 40.)

[Van der Aalst 2011c] Wil MP Van der Aalst and Wil Van der Aalst. Process mining: discovery, conformance and enhancement of business processes. Springer, 2011. (Cited on pages 1, 2, 10, 14, 19, 39 and 52.)

[Van der Aalst 2012] Wil Van der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs*et al. Process mining manifesto.* In Business process management workshops, pages 169–194. Springer, 2012. (Cited on page 10.)

[Van der Werf 2008] Jan Martijn EM Van der Werf, Boudewijn F van Dongen, Cor AJ Hurkens and Alexander Serebrenik. *Process discovery using integer linear programming.* In Applications and Theory of Petri Nets, pages 368–387. Springer, 2008. (Cited on page 22.)

[Van der Werf 2011] J.M.E.M. Van der Werf. *Compositional Design and Verification of Component Based Information Systems.* 2011. (Cited on pages 1 and 2.)

[van Dongen 2004a] Boudewijn F van Dongen and Wil MP Van der Aalst. *EMiT: A process mining tool.* In Applications and Theory of Petri Nets 2004, pages 454–463. Springer, 2004. (Cited on pages 28, 29, 31 and 40.)

[van Dongen 2004b] Boudewijn F van Dongen and Wil MP Van der Aalst. *Multi-phase process mining: Building instance graphs.* In Conceptual Modeling–ER 2004, pages 362–376. Springer, 2004. (Cited on pages 20, 21, 24, 29 and 40.)

[van Dongen 2005a] Boudewijn F van Dongen, Ana Karla A de Medeiros, HMW Verbeek, AJMM Weijters and Wil MP Van der Aalst. *The ProM framework: A new era in process mining tool support.* In Applications and Theory of Petri Nets 2005, pages 444–454. Springer, 2005. (Cited on page 29.)

[van Dongen 2005b] Boudewijn F van Dongen and Wil MP Van der Aalst. *A Meta Model for Process Mining Data.* EMOI-INTEROP, vol. 160, 2005. (Cited on pages 28, 30 and 40.)

[van Dongen 2005c] Boudewijn F van Dongen and Wil MP Van der Aalst. *Multi-phase process mining: Aggregating instance graphs into EPCs and Petri nets.* In Proceedings of the 2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management (PNCWB). Citeseer, 2005. (Cited on pages 28, 29 and 40.)

[van Dongen 2005d] Boudewijn F van Dongen, Wil MP Van der Aalst and Henricus MW Verbeek. *Verification of EPCs: Using reduction rules and Petri nets.* In Advanced Information Systems Engineering, pages 372–386. Springer, 2005. (Cited on page 41.)

[Van Glabbeek 1996] Rob J Van Glabbeek and W Peter Weijland. *Branching time and abstraction in bisimulation semantics.* Journal of the ACM (JACM), vol. 43, no. 3, pages 555–600, 1996. (Cited on page 24.)

[Van Lamsweerde 2001] Axel Van Lamsweerde. *Goal-oriented requirements engineering: A guided tour.* In Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on, pages 249–262. IEEE, 2001. (Cited on pages 32 and 33.)

[Veblen 1898] Thorstein Veblen. *Why is economics not an evolutionary science?* The Cambridge, 1898. (Cited on pages 2 and 3.)

[Weijters 2001] AJMM Weijters and WMP Van der Aalst. *Process mining: discovering workflow models from event-based data.* In Proceedings of the 13th

Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001), pages 283–290, 2001. (Cited on pages 22, 24, 27, 28, 30 and 41.)

[Weijters 2003] Anton JMM Weijters and Wil MP Van der Aalst. *Rediscovering workflow models from event-based data using little thumb.* Integrated Computer-Aided Engineering, vol. 10, no. 2, pages 151–162, 2003. (Cited on pages 20, 22, 24, 27, 29 and 40.)

[Wen 2006] Lijie Wen, Jianmin Wang and Jiaguang Sun. *Detecting implicit dependencies between tasks from event logs.* In Frontiers of WWW Research and Development-APWeb 2006, pages 591–603. Springer, 2006. (Cited on page 21.)

[White 2004] Stephen A White. *Introduction to BPMN.* IBM Cooperation, vol. 2, no. 0, page 0, 2004. (Cited on page 23.)

[Witten 2005] Ian H Witten and Eibe Frank. Data mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2005. (Cited on pages 36 and 39.)

[Xing 2006] Zhenchang Xing and Eleni Stroulia. *Refactoring practice: How it is and how it should be supported-an eclipse case study.* In Software Maintenance, 2006. ICSM'06. 22nd IEEE International Conference on, pages 458–468. IEEE, 2006. (Cited on page 118.)

[Yi 2007] Ji Soo Yi, Youn ah Kang, John T Stasko and Julie A Jacko. *Toward a deeper understanding of the role of interaction in information visualization.* Visualization and Computer Graphics, IEEE Transactions on, vol. 13, no. 6, pages 1224–1231, 2007. (Cited on page 42.)

[Yin 2009] Robert K Yin. Case study research: Design and methods, volume 5. sage, 2009. (Cited on page 6.)

[Yu 1987] Eric SK Yu. *What Does It Mean to Say that a Specification is Complete?* In Software Specification and Design, Proceeding of IWSSD-4, Fourth International Workshop on. IEEE, 1987. (Cited on pages 7, 32 and 33.)

[Yu 2011] Eric Yu. *Modelling strategic relationships for process reengineering.* Social Modeling for Requirements Engineering, vol. 11, page 2011, 2011. (Cited on pages 6, 37 and 45.)

[Zave 1997] Pamela Zave and Michael Jackson. *Four dark corners of requirements engineering.* ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 6, no. 1, pages 1–30, 1997. (Cited on page 33.)

[Zhang 2003] Shao-hua Zhang, Ning Gu, Jie-Xin Lian and Sai-Han Li. *Workflow process mining based on machine learning.* In Machine Learning and Cybernetics, 2003 International Conference on, volume 4, pages 2319–2323. IEEE, 2003. (Cited on pages 28, 29 and 41.)

[Zhang 2004] Yingjian Zhang. *Prediction of financial time series with Hidden Markov Models*. PhD thesis, Simon Fraser University, 2004. (Cited on page 50.)

[Zoukar 2005] Iyad Zoukar. *MIBE: Méthode d'Ingénierie des Besoins pour l'implantation d'un progiciel de gestion intégré (ERP)*. PhD thesis, Paris 1, 2005. (Cited on page 17.)

[Zur Muehlen 2000] Michael Zur Muehlen and Michael Rosemann. *Workflow-based process monitoring and controlling-technical and organizational issues*. In System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on, pages 10–pp. IEEE, 2000. (Cited on page 31.)

Abstract

Abstract: So far, process mining techniques suggested to model processes in terms of tasks that occur during the enactment of a process. However, research on process modeling has illustrated that many issues, such as lack of flexibility or adaptation, are solved more effectively when intentions are explicitly specified. This thesis presents a novel approach of process mining, called Map Miner Method (MMM). This method is designed to automate the construction of intentional process models from traces. MMM uses Hidden Markov Models to model the relationship between users' activities and the strategies (*i.e.*, the different ways to fulfill the intentions). The method also includes two specific algorithms developed to infer users' intentions and construct intentional process model (Map), respectively. MMM can construct Map process models with different levels of granularity (pseudo-Map and Map process models) with respect to the Map metamodel formalism. The entire proposed method was applied and validated on practical traces in a large-scale experiment, on event logs of developers of **Eclipse UDC** (Usage Data Collector). The resulting Map process models provide a precious understanding of the processes followed by the developers, and also provide feedback on the effectiveness and demonstrate scalability of MMM in terms of traces. Map Miner tool has been developed to enable practicing the proposed approach. This permits users to obtain the pseudo-Map and Map process model out of traces.

Keywords: Intentional Process Models, Machine Learning, Process Mining, Hidden Markov Models

Résumé : Jusqu'à présent, les techniques de fouille de processus ont modélisé les processus en termes des séquences de tâches qui se produisent lors de l'exécution d'un processus. Cependant, les recherches en modélisation du processus et de guidance ont montré que de nombreux problèmes, tels que le manque de flexibilité ou d'adaptation, sont résolus plus efficacement lorsque les intentions sont explicitement spécifiées. Cette thèse présente une nouvelle approche de fouille de processus, appelée Map Miner méthode (MMM). Cette méthode est conçue pour automatiser la construction d'un modèle de processus intentionnel à partir des traces d'activités des utilisateurs. MMM utilise les modèles de Markov cachés pour modéliser la relation entre les activités des utilisateurs et leurs stratégies (*i.e.*, les différentes façons d'atteindre des intentions). La méthode comprend également deux algorithmes spécifiquement développés pour déterminer les intentions des utilisateurs et construire le modèle de processus intentionnel de la Carte. MMM peut construire le modèle de processus de la Carte avec différents niveaux de précision (pseudo-Carte et le modèle du processus de la carte) par rapport au formalisme du métamodèle de Map. L'ensemble de la méthode proposée a été appliqué et validé sur des ensembles de données pratiques, dans une expérience à grande échelle, sur les traces d'événements des développeurs de **Eclipse UDC**.

Les modèles de processus obtenus fournissent une compréhension précieuse des processus suivis par les développeurs, et fournissent également des informations sur l'efficacité et démontrent l'évolutivité de MMM. L'outil de Map Miner tool a été développé pour permettre la pratique de l'approche proposée. Cela permet aux utilisateurs d'obtenir la pseudo-Carte et le modèle du processus de la Carte à partir des traces.

Mots Clés : Modèles de processus intentionnels, Apprentissage automatique, Fouille de processus, Modèles de Markov cachés